

# **The Development of Computational Thinking Skills in Young Children**

A Dissertation Submitted by

Emily Vera Relkin

In partial fulfillment of the requirements for the degree of Doctor of Philosophy

in *Child Study and Human Development*

TUFTS UNIVERSITY

June 10, 2022

Committee Members:

Marina U. Bers, Ph.D. (chair)

Eliot-Pearson Department of Child Study & Human Development, Tufts University

Sara K. Johnson, Ph.D.

Eliot-Pearson Department of Child Study & Human Development, Tufts University

Brian E. Gravel, Ph.D.

Department of Education, Tufts University

Gregory K.W. K. Chung, Ph.D.

National Center for Research on Evaluation, Standards, and Student Testing (CRESST), UCLA

## Abstract

This study explores how children from two to nine years of age develop Computational Thinking (CT) skills. The term “CT” denotes a set of cognitive processes that are useful for framing and solving problems using computers and other information processing agents. Acquisition of CT skills is traditionally associated with learning to code but other factors may influence accrual in the course of normal development. To explore this issue, I examined two- to nine-year-old children’s performance on four grade-specific versions of the 15-item *TechCheck* “unplugged” assessment probing six domains of CT. I collected data from coding-naive children receiving the version of *TechCheck* designed for their grade and compared their performance to that of coding-naive students in one grade higher who were administered the same *TechCheck* version. *TechCheck* scores in all grades were normally distributed. Mean scores were significantly greater in students in the higher of each grade pair. This finding suggests that average performance in solving unplugged problems that probe CT skills may improve with advancing grade in the absence of coding instruction. Linear mixed modeling identified grade and the interaction of grade with age as predictors of *TechCheck* performance in coding-naive students. Next, I examined *TechCheck* data obtained from first graders before and after they received one of three coding educational interventions. *TechCheck* scores improved significantly after children learned to code compared to a non-coding control group, providing evidence that the acquisition of CT can be accelerated by coding education. Finally, I explored whether grade and coding interventions differentially affected performance across CT domains. With each advancing grade, coding-naive students scored higher in all six CT domains evaluated. Children who were taught to code showed more selective improvements in specific CT domains depending on which coding educational intervention they received. Limitations to this study include the use of heterogeneous cohorts drawn from multiple studies and the exploratory nature of domain analysis using *TechCheck*. I conclude that CT skills can improve in the course of early childhood without CS instruction, perhaps as a result of learning from everyday experiences, non-CS education, and/or brain maturation. Learning to code at a young age, which is known to foster improved communication and creative self-expression, can accelerate the acquisition of CT but may have more selective effects on specific CT domains depending on how coding is taught.

### *Keywords*

Assessment, Computer Science, Coding, Early Childhood, Child Development

## Acknowledgements

This dissertation would not have been possible without those who have provided me with immense support throughout my Ph.D. process.

First and foremost, to my wonderful advisor, Dr. Marina Bers. I am extremely fortunate to have had you as my mentor. One of the best decisions I ever made was to volunteer at your lab six years ago. Despite knowing absolutely nothing about the field at this time, I saw how special the environment you had cultivated at DevTech was and I wanted nothing more than to be a part of it. As soon as I became your Ph.D. student I realized I had found my academic niche. I'm sure it wasn't always easy mentoring me, especially in the beginning when I needed extra support to get caught up to speed. You have consistently gone above and beyond for me and have provided me countless invaluable learning opportunities that I never imagined I would have. It has been truly an honor to be your student and to be consistently inspired, guided, and supported by your leadership. You have had such a profound impact on both my personal and academic development and I want you to stay my mentor and a big part of my life forever! I'll be sure to bring everything I have learned from you with me wherever I go next.

I truly believe I have the best possible Ph.D. committee. Dr. Sara Johnson is one of the most caring and generous professors I have ever had. I cannot thank you enough for your feedback and advice on my work throughout my Ph.D. and on my dissertation. The way that you mentor your students is really something I admire. Dr. Brian Gravel was also an invaluable member of my Master's thesis committee. Your thoughts and feedback really molded *TechCheck* into a better assessment. Thank you for completely changing my understanding of computational thinking and research methods in general and continuing to push me to think in new ways. Thank you to Dr. Gregory Chung who has been my biggest advocate and supporter, both with my

dissertation work and in my outside work at UCLA CRESST over the last two years. It has been such a privilege to have had the opportunity to work alongside and learn from you and your brilliant team. I hope I can continue to do so for many years to come.

I am deeply grateful to my family members who have provided me with unconditional support throughout this process. My two incredible parents, my father, Dr. Norman Relkin, and my mother, Dr. Felicia Relkin, both deserve their own honorary Ph.D. in Child Study and Human Development for the immense support they have both given me not only during this process but also through the hard work of raising me. Thank you for both pushing me to pursue a Masters and Ph.D. in the first place. Thank you for the copious amounts of phone calls in which you were asked to proofread, edit, practice presentations, or give research advice. Thank you in particular for the moral support during the occasional late-night panicked phone call from me, such as the time I spilled coffee on my computer and lost my 34-page final the day before it was due. I truly couldn't have done any of it without you. To my older brother, Paul Relkin, the family computer scientist, thank you for always trying to teach me to code, hack, or play with robots when we were kids and sorry that I was never interested in learning. As you can see from this dissertation you finally succeeded in sparking my interest in this topic.

To my significant other, Dr. Myles Dworkin, you do the most amazing job of balancing stress and lack of sleep due to life as a surgical resident while still providing me with immense amounts of support in so many ways. Your dedication to your profession and research is incredibly admirable and something that pushes me to also do better every day. I particularly appreciate the cakes and celebrations that were provided after each Ph.D. milestone (shout out to Sivan and Rosie Adler because we all know he didn't have time during work and they were the



real ones to pull that off). Thank you also to Myles' parents and siblings who have welcomed me into their family and have taken such good care of the two of us during these eventful years.

My sincere appreciation goes to the best team of researchers I could have asked for at the DevTech Research Group. Each of you has had such an impact on my life! A special thanks to Aim Unahalekhaka, Dr. Amanda Strawhacker, Dr. Madhu Govind, Dr. Amanda Sullivan, Dr. Jan Yang, and Tess Levinson. Thank you to Dr. Laura de Ruitter who taught me how to analyze data with R! The wonderful lab managers Riva Dhamala, Anne Drescher, and Jessica Blake-West were so important in holding everything at the lab together and helping me stay on top of my work. I also want to thank the following undergraduate and graduate students who helped with these projects: Elizabeth Hunt, Melissa Veizel, Dr. Ziva Hassenfeld, Maya Morris, Jaclyn Tsiang, Ari Lerner, Brendan Brennan, Hasan Khan, Alexa Hasse, Patrick Nero, Megan Bennie, Rachel Viselman, Kaylyn Adams, and Hannah Riehl. Each of you made significant contributions to this work, and I could not have gotten to this point without you.

Thank you to the members of the CRESST team at UCLA who have supported me in various ways. Dr. Markus Iseli has been such a pleasure to work with and has helped me understand the definition of computational thinking more comprehensively through natural language processing. I'll be sure to visit you in Switzerland! Dr. Yon Soo Suh is an incredible research scientist and psychometrician and has taught me so many new statistical techniques. Teanna Feng is one of the most dedicated and skilled Ph.D. students I've come across.

The Norfolk study project coordinator, Angela de Mik, I think, had the toughest job on the project. Thank you for the immense amount of work you did on the ground to lead all of the educators and students. You did whatever it took to help the teachers and students, even if that meant jumping in and teaching the kids yourself! Thank you to the DOE study project

coordinators Parastu Dubash and Megan Bennie for your amazing organization skills and willingness to help and answer any questions I had throughout these projects. This work would not be possible without the Norfolk Public School ITRTs, The Shaffer Evaluation Group, and the DevTech Assessment Team.

Thank you to Mika Fifi Relkin-Dworkin, the best cat I could have ever asked for. Thank you for keeping me happy and giving me the best company while I wrote this dissertation.

Last but not least, I want to sincerely thank all of the educators, students, and parents who made this project possible. My greatest hope is that this work can give back to them by supporting young children's learning and development.

This dissertation research utilized data from various studies generously supported by the following grants:

Department of Defense Education Activity (DoDEA) through Grant "Operation: Break the Code for College and Career Readiness", Unique Entity Identifier: "WORLDCL10" awarded to the Norfolk, Virginia Public Schools.

Education Innovation and Research, The U.S. Department of Education Through grant PR/Award Number: U411C190006 awarded to the Tufts University DevTech Research Group.

The Institute of Education Sciences, U.S. Department of Education, through Grant Award Number: R305A190433 awarded to the University of California, Los Angeles.

LEGO Foundation through Grant "Supporting and Amplifying Local Organizations Engaged in Playful Engineering-Based Learning Post-COVID" awarded to the Tufts University DevTech Research Group

## TABLE OF CONTENTS

Abstract.....	2
Acknowledgements.....	3
Introduction.....	8
Statement of the Problem.....	12
Research Questions.....	14
Background.....	14
A Brief History of CT.....	14
Defining CT Domains.....	20
Powerful Ideas of CT for Early Childhood.....	22
CT Assessment in Young Children.....	25
Unplugged CT Assessment.....	27
The <i>TechCheck</i> CT Assessment.....	29
Method.....	32
Experimental Design.....	32
Data Sources.....	34
Coding Interventions.....	35
Coding Curricula.....	37
CT Assessment Administration.....	39
Data Analysis.....	41
Results.....	43
RQ1.....	43
RQ2.....	54
RQ3.....	57
Discussion.....	65
Limitations and Future Work.....	75
Conclusions.....	81
References.....	84
Appendix.....	105

## Introduction

In today's digital world, technologies play an increasingly important role in nearly every child's life and development. In many countries, children begin to use smart technologies before the age of two (AVG Technologies, 2018; Gerson, et al., 2022; OECD, 2010). Children's access to technology, their technological engagement, and their family's perceptions of technologies can have a significant impact on their developmental trajectory (Hartle, 2019). There is an evolving need to align education with the socio-digital revolution that has impacted our lives and educational systems so pervasively (Ezeamuzie & Leung, 2021).

Accordingly, one of the goals of computer science (CS) education is to foster the development of a class of thought processes that facilitate problem solving with computers and other technologies; these processes are known as "Computational Thinking" (CT). The acquisition of CT skills is considered a vital goal of CS education because it can provide the necessary cognitive framework for succeeding in today's technology-driven society. CT skills can enhance coding ability and promote problem-solving in other disciplines (Barr & Stephenson, 2011; Bers, 2020; Chen et al., 2017; Cuny et al., 2010; Wing, 2006).

The precise definition of CT remains the subject of controversy. The definition has been debated, as has the question of whether CT represents a singular unified concept or is multifaceted (Barr & Stephenson, 2011; Grover & Pea, 2013; National Research Council, 2011). For present purposes, CT can be broadly defined as a set of skills and processes that can be used to represent and solve problems in a form that can be carried out by an information-processing agent (human or machine) (Iseli et al., 2022; Wing, 2006, 2011). Further discussion of the history of CT, its scope, and constituent domains is presented in the Background section below.

Until recently, most research on CT did not take into account the context of early childhood. To address this gap, Bers (2018) reviewed the literature and identified twenty powerful ideas of CT and CS that could be considered developmentally appropriate for young children. The twenty ideas were based on frameworks such as Brennan and Resnick (2012), Google for Education (2010), and years of research with the KIBO robotics kit and the ScratchJr coding application (e.g., Portelance & Bers, 2015; Strawhacker et al., 2017; Sullivan et al., 2017; Sullivan & Bers, 2017). Those ideas were ultimately distilled down to seven powerful ideas that are developmentally appropriate for early childhood CS education. They include the domains of hardware/software, algorithms, modularity, control structures, representation, debugging, and design process.

Once the importance of CT was recognized in the context of CS education, a need arose for reliable and valid methods of assessing CT. The majority of validated CT assessments are designed for older children and adults (Chen et al., 2017; Fraillon et al., 2018; Román-González et al., 2018; Werner et al., 2012). Very few CT assessments have been created for early elementary school children and fewer have been tested with a substantial number of children to confirm they have acceptable psychometric properties (Santos et al., 2020). The lack of a gold-standard assessment of CT in early childhood made it very difficult to measure young children's progress in learning or gauge the effectiveness of CS lessons. The paucity of suitable measures has been identified as an obstacle towards integrating CS education into early elementary school settings (Lee et al., 2011; Lockwood & Mooney, 2018; Román-González et al., 2019).

When I began my graduate studies in 2016, perspectives on children's CT development either came from small observational/interview-based studies in young children (e.g., Mioduser,

et al., 2009; Portelance & Bers, 2015) or were extrapolations from studies carried out in older children and adults (e.g., Basu et al., 2016; Werner et al., 2014). At the time, the assessments that did exist for early elementary school students were qualitative and typically required expert scoring and administration. Instruments for assessing CT in older students and adults have existed for some time and typically involved coding challenges requiring some familiarity with a programming language (Chen et al., 2017; Fraillon et al., 2018; Werner et al., 2012). However, coding-based CT instruments are subject to a floor effect in coding-naive students, which largely precludes their use as a baseline measure in longitudinal studies. Furthermore, such measures potentially conflate coding and CT skills (Yadav, et al., 2017a).

My initial venture into the area of CT assessments for young children occurred while carrying out research towards my Master's degree. It involved my creation of an adaptive, interview-based assessment of CT and coding called "Tufts Assessment of Computational Thinking in Children - KIBO Robot Version" (TACTIC-KIBO) (Relkin, 2018). This instrument used the KIBO robot programming platform and employed coding challenges of increasing complexity to evaluate CT proficiency in four to nine-year-old-children. TACTIC-KIBO drew upon Bers (2018)'s Seven Powerful Ideas of CS as a basis for developmentally appropriate assessment. Scores on this instrument showed good inter-rater reliability and content validity when compared to expert ratings. However, TACTIC-KIBO suffered from the same scalability issues, floor effects, and conflation of coding with CT as other coding platform-specific CT assessment instruments.

By the time I started my doctoral studies, it was clear that a new approach was needed to measure CT in young children. A major challenge was finding an approach that was free of any dependency on a particular coding platform. In 2018, my advisor, Dr. Marina Bers, asked me to

create a version of TACTIC-KIBO that could be used in a large-scale study involving two different coding educational platforms (KIBO and code.org). Because the instrument needed to be administered to hundreds of students in classroom settings, I transformed TACTIC-KIBO from an interview/play-based format into a multiple-choice assessment. Given the impracticality of creating new versions of TACTIC every time a new programming platform was encountered, Dr. Bers had the foresight to suggest that I start thinking about a future iteration of this assessment that could be platform-independent.

The exercise of designing a version of TACTIC for the code.org platform ultimately provided me with the inspiration for creating a platform-independent CT instrument, in line with Dr. Bers' suggestion. The inspiration came from the fact that the version of TACTIC for code.org contained a few questions that can be described as “unplugged” challenges. Unplugged activities have been used in CS education for many years as a means of invoking the principles of CS and/or CT without requiring children to have any knowledge of programming (Bell & Vahrenhold, 2018). Unplugged activities typically involve puzzles, games, and exercises that are well known to most K-12 children and exemplify CS concepts without requiring knowledge of coding or the use of computers. These activities have been used to teach CS concepts for over two decades (e.g., CSUnplugged.com; code.org). Realizing that unplugged activities could potentially be used to resolve the dependency on coding exercises used in previous instruments, I set about creating a new type of CT assessment for young children.

I designed the *TechCheck* assessment to measure CT in early childhood in a format suitable for use in general classrooms as well as large-scale research studies (Relkin et al., 2020, 2021; Relkin & Bers, 2021). *TechCheck* presents children with engaging puzzle-like challenges that are analogous to those that arise in the course of computer programming but do not require

computers or coding experience to complete (Relkin et al., 2020, 2021; Relkin & Bers, 2020). In 2018-2019, I field-tested *TechCheck* in Boston area schools to establish its age-appropriateness and criterion validity. *TechCheck* was then included among the instruments administered to hundreds of kindergarten, first, and second grade students taking part in a study in Norfolk, Virginia of a computer coding curriculum called “Coding as Another Language” (CAL). As a consequence, *TechCheck* then became one of the first unplugged assessments of CT for young children that had evidence of validity and reliability obtained from a large sample of students. *TechCheck* was subsequently shown to be sensitive to change when applied longitudinally with or without a coding intervention (Relkin et al., 2021).

Originally, I created a single version of *TechCheck* for kindergarten through second grade. Initial experience with the assessment indicated that CT skills changed sufficiently over this age range to warrant the creation of grade-specific versions. At the time of this writing, there are four versions of *TechCheck* in use (PreK, K, Grade 1, and Grade 2). Although all of these versions have been tested fairly extensively, only limited analyses have been performed across studies. The availability of data on young children’s CT skills obtained from multiple studies employing *TechCheck* presents a unique opportunity to explore how CT develops and how CS education influences its development. Ways in which CT education and assessment can be improved, in turn, may be suggested.

### **Statement of the Problem**

Much of the current understanding about how children develop CT skills has been closely tied to coding and CS education. There is little information available about how other processes such as brain maturation, everyday interactions with technology, non-technological experiences (including everyday problem solving), and educational interventions influence the ability of



young children to engage and master CT concepts. Using unplugged assessment, it is now possible to study how these and other factors influence the acquisition and mastery of CT. This recognition provides the motivation for the exploratory studies described in this dissertation.

An important question that has been largely unexplored in the childhood CT literature is the extent to which CT skills accrue incidentally in the course of normal development versus being acquired through CS education. There is an accepted relationship between learning to code and the development of CT skills (Arfé et al., 2020; Lodi, 2020). However, there is little information available on whether CT can be acquired in the absence of learning to code, and if so, how that compares to CT skills acquired through coding and other CS education.

## Figure 1

### *CT and Its Relationship to Young Children's Everyday Life*

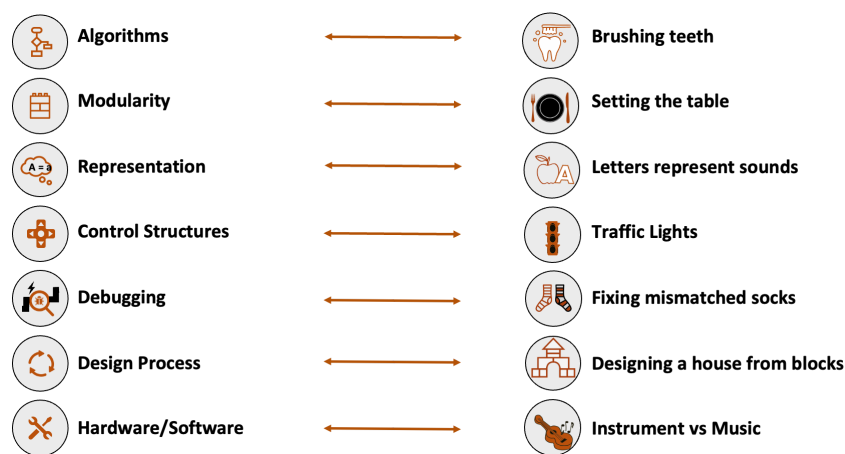


Figure 1 lists examples of CT-related concepts that have analogies to activities that young children typically engage in during the course of their everyday lives. It seems possible that children can learn aspects of the reasoning involved in CT through these and other routine experiences without necessarily having exposure to coding or other CS instruction. Likewise, brain development throughout childhood could contribute to improvements in the various

cognitive functions that subserve CT skills, exclusive of learning to code (Arfé, et al., 2020). This is a testable hypothesis, to the extent that one would expect at least some coding-naive children to score well on unplugged CT assessments and for performance to improve with advancing age/grade/experience.

### **Research Questions**

In this dissertation, I explore how children develop CT skills in early childhood through the following research questions:

**RQ1.** To what extent do CT skills differ in coding-naive children in preschool through third grade?

**RQ2.** Which demographic and environmental factors predict baseline CT performance in children ages 2-9?

**RQ3.** How do coding educational interventions (specifically CAL-ScratchJr, CAL-KIBO, and codeSpark) affect the rate of acquisition of CT skills overall and in select subdomains?

## **Background**

### **A Brief History of CT**

Although the past two decades have seen a precipitous rise in interest in CT, it is not an entirely new concept or term. The word “computation” comes from the Latin root “computationem” which originally meant an “act, process, or method of arithmetic calculation”. Earliest recorded use of the word “computational” reportedly dates to 1857 when it was taken to mean, “pertaining to or of the nature of a computation” (etymonline.com). A few years prior to the emergence of the term “computational”, Charles Babbage invented his “Difference Engine”, which is often cited as the predecessor of modern programmable computers (Babbage, 1832; Swade, 2005). In the 1940s, the term “computational thinking” was used to describe the types of thought processes involved in projecting and quantifying future needs (Prakken, 1942). In this

context, “computational thinking” represented the thought processes one might use (perhaps aided by a slide rule or an adding machine) to estimate future tax payments or calculate projected travel expenses (The Mathematics Teacher, 1943).

It is important to clearly distinguish CT from terms such as “computing”, “Computer Science (CS)” and “coding”, which are sometimes used interchangeably in ways that blur the boundaries between them (Zhang & Nouri, 2019). “Computing” is a broad term that includes CS, computer and software engineering, information technology, data science, and information systems (Denning, 2007). CS is the study of computation. It is a discipline that includes artificial intelligence, graphics, virtual reality, databases, and other branches (Barr & Stephenson, 2011). Thus, CS is not limited to coding or CT, nor is CT exclusively a part of computer science as it is applicable to a variety of fields and can be involved in everyday life activities (Li et al., 2020; Relkin & Strawhacker, 2021). Coding (programming) is a technical skill that is described as “the instrumental skill of CT” and is currently considered “the primary means of teaching CT in primary school” (Arfé et al., 2020; Relkin et al., 2021; Román-González, et al., 2017; Wing, 2006). Coding is considered a part of CS and essential for all computer scientists to learn (Ezeamuzie & Leung, 2021). CT skills may be acquired without the use of coding, for example through unplugged activities (Bell et al., 2009; Hermans & Aivaloglou, 2017; Metin, 2020; Wohl et al., 2015). Although CT is essential for programming computers, coding can be carried out without CT. For example, coding using a copy-and-paste approach or rote memorization of syntax could be considered programming without invoking CT (Bers, 2018; Bortz et al., 2019).

In reviewing the history of CT, at least three periods can be identified in which the associated concepts underwent progressive development and refinement. I will refer to these as the “Foundational,” Transitional” and “Modern” periods, respectively.

## 1. Foundational Period of CT

The foundations of modern conceptions of CT can be traced back to the 1960s. Alan Perlis was one of the first computer scientists to advocate for the inclusion of computer programming in higher education. He argued “One of the problems is that the pedagogy for computers hasn’t yet been developed properly” (Katz, 1960, p. 522). Knuth (1974), another well-regarded computer scientist and mathematician, echoed this notion and wrote about how using an algorithmic approach to programming can deepen understanding of concepts in many disciplines. The computer scientist Edsger Dijkstra (1979) wrote about computational habits that aided in successful computer programming. These habits included separation of concerns, effective use of abstraction, the design and use of notations tailored to one's manipulative needs, and avoiding case analyses.

One of the first uses of the actual term “CT” in relation to computer science was by Seymour Papert in his book *Mindstorms* (Papert, 1980). Papert made a brief reference to CT in the context of discussing the challenge of integrating CS education with children’s everyday experiences. Papert briefly mentioned CT again in reference to the expertise of the scientists who first used computers to calculate missile trajectories (Papert, 1993). He later used the term in a discussion of methods for solving geometric problems in which he contrasted the use of a Monte Carlo simulation with the application of Euclidean geometry tools. He made the point that computational solutions could foster a better understanding of underlying mathematical concepts (Papert, 1996). In these and other publications, Papert’s concept of CT is tied to Constructionism, which is the principle of epistemology that posits that knowledge can be acquired by the physical manipulation/ programming of objects such as computers and other technological tools (Papert & Harel, 1991). Papert had added to Piaget’s Constructivist theory by taking into account the

environment, artifacts, and individual decisions involved in constructing knowledge. Papert's conception of CT includes the principle that social and emotional involvement with technology can make programming an effective tool for learning other disciplines (Lodi & Martini, 2021). Papert used the term "powerful ideas" more frequently than "computational thinking" in referring to concepts that afford new ways of thinking and applying knowledge (Bers, 2017; Papert, 2000).

## 2. Transitional Period of CT

diSessa (2000) wrote about the concept of "computational literacy," a term that overlaps with modern definitions of CT. Both CT and computational literacy highlight the importance of engaging in computation for the development of certain aspects of cognition. diSessa distinguished between *computer literacy*, which primarily involves familiarity with the operation of computers, and *computational literacy*, which he likened to textual literacy in its breadth and importance. In more recent publications, diSessa (2018) and Li et al., (2020) have argued that computational literacy is a broader concept than CT because it emphasizes the social and cultural aspects of computing and not just the cognitive impact of learning to code.

Wing's seminal article popularized CT as a fundamental skill set useful for solving problems, designing systems, and understanding human behavior (Wing, 2006). She argued that "everyone could benefit from thinking like a computer scientist" and that CT should be added to every child's education just like reading, writing, and arithmetic. Cuny et al. (2010) later redefined CT as "...the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (p. 1). Wing's perspective on CT differed from that of Papert and other CS scholars that preceded her (Denning, 2017). Papert emphasized the value of knowledge

and intuitions built from interactive experiences with technological tools. Wing's definition placed more of an emphasis on the concepts associated with programming and less on learning from experience through exposure to technological tools. Wing (2008) argued:

We also do not want people just to be able to use the tool but not have learned the concepts (a case in point: using a calculator versus understanding arithmetic). Worse, we do not want people to come away thinking they understand the concepts because they are adept at using the tool (p 3721).

Barba (2016) notes that CT as proposed by Wing (2006) does not embody many of the main ideas Papert (1996) proposed. Wing's formulation is contrary to Papert's "Power Principle" which is the idea that students should first experience computational tools and later construct/understand the associated concepts. Papert recognized when concepts are taught out of context they can become disempowered and no longer personally meaningful to the student. An editorial by Denning (2017) summarized the difference between "Traditional CT" (as envisioned by Papert, Knuth, Dijkstra, etc) and "New CT" (as followed from Wing's seminal 2006 article). (Table 1). Denning notes that in "Traditional CT" the process of coding leads to the acquisition of CT, whereas in "New CT" learning CT concepts tends to promote programming skills.

### **3. Modern Period of CT**

Inspired by the groundwork of Papert, diSessa, Wing, and others, the Modern period has produced a plethora of definitions and categorizations of CT. The operationalization of CT helped establish it as a core competency of CS education. The lack of consensus about the definition of CT may be attributed to the diverse backgrounds and perspectives of the defining experts and their varied motivations (e.g., educational, research, policy). Another factor is changes in affordances as programmable technologies evolve.

**Table 1***Differences between “Traditional” and new CT*

<b>Traditional CT</b>	<b>New CT</b>
Mental habits and disciplines for designing useful software	Formulating problems so that their solutions can be expressed as computational steps
Extensively practicing programming cultivates CT as a skill set	CT is a conceptual framework that enables programming
Skills of design and software crafting—for example separation of concerns, effective use of abstraction, devising notations tailored to one’s needs, and avoiding combinatorically exploding case analyses	Set of problem solving concepts such as representation, divide-and-conquer, abstraction, information hiding, verification, and logical reasoning
A new way of conducting science, alongside theory and experiment—a revolution in science	Useful in sciences and most other fields
Algorithms are directions to control a computational model (abstract machine) to perform a task	Algorithms are expressions of recipes for carrying out tasks; no awareness of computational models is needed
Programs are tightly coupled with algorithms; programs are algorithms expressed in a computer language; algorithms derive their precision from a computational model	Programs are loosely coupled with algorithms; algorithms are for all kinds of information processors including humans—it is completely optional whether an algorithm will ever be translated into a program
Designing computations in a domain requires extensive domain knowledge	Someone schooled in the principles of CT can find computational solutions to problems in any domain
End users can follow algorithms and get the result without any understanding of the mechanism	People engaging in any step-by-step procedure are performing algorithms and are (perhaps unconsciously) thinking computationally
Engaging in a computational task without awareness is not computational thinking	People who are engaging in any task that could be performed computationally are engaging in subconscious computational thinking

*Note:* This table is from *Denning (2017)*

CT is now characterized in many different ways. Many researchers agree that CT can be broadly and generically defined in ways that focus on universal problem-solving skills (Lodi, 2020; Tikva & Tambouris, 2021; Zhang & Nouri, 2019). CT has also been operationalized as a multifaceted concept, practice, or competency that includes domains such as sequences, loops, algorithmic thinking, debugging, parallelism, and events (Brennan & Resnick, 2012; Ezeamuzie & Leung, 2021; Tikva & Tambouris, 2021; Zhang & Nouri, 2019). Iseli et al. (2022) further broke down these concepts into emergent constructs (that arise from CT), contributing constructs (that contribute to CT), and coding concepts (directly related to computer programming). Some

scholars define CT as “transversal” or allowing for the transfer of knowledge and expansion of CT beyond computer science and programming (Li et al., 2020; Lodi, 2020). CT has been recognized as having psychological effects that change views students form about the world around them and about themselves (Brennan & Resnick, 2012; Li et al., 2020). The majority of definitions relate CT to coding in some way, as well as with more general problem-solving skills.

During the “modern” period of CT, various scholars tried to define the domains underlying CT as part of establishing it as a core competency in K-12 education. An even clearer tendency to connect coding and CT concepts is evident in the categorization of CT domains, described below.

### Defining CT Domains

The plethora of CT domain definitions in the literature has made it difficult to standardize and compare studies and methods of teaching (Ezeamuzie & Leung, 2021). A summary of the domains in major papers within the CT literature can be found in Table 2.

**Table 2**

*CT domains that have been identified by key authors*

	Wing (2006)	NRC (2010)	Barr & Stephenson (2011)	ISTE & CSTA (2011)	ISTE & CSTA Standards (2011)	Grover & Pea (2013)	Selby & Woollard (2013)	Kalelioğlu et al. (2016)	Shute, et al. (2017)	Bers (2018)	Google (N.D.)	CS Unplugged (N.D.)	total
Algorithms	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	10
Decomposition/Modularity			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10
Abstraction			✓	✓	✓	✓	✓				✓	✓	8
Simulation/Modeling	✓	✓	✓	✓				✓	✓		✓		7
Patterns/Generalization					✓		✓	✓	✓		✓	✓	6
Representation		✓	✓	✓		✓				✓	✓		6
Debugging	✓	✓				✓		✓	✓	✓			6
Data Processing		✓	✓	✓				✓	✓		✓		6
Parallelism	✓	✓		✓		✓		✓			✓		6
Control Logic/Conditionals	✓			✓		✓				✓		✓	5
Automation/Hardware & Software			✓	✓				✓		✓	✓		5
Evaluation					✓		✓					✓	3
Creative Design		✓								✓			2
Iteration						✓			✓				2



Wing (2008) subsequently refined her previously broad statements by identifying certain domains of CT. She emphasized that abstraction and automation were the core concepts of CT. In 2010, the National Research Committee conducted a workshop to discuss how to best introduce CT to students (NRC, 2010). Wing's (2006,2008) statements about CT were expanded with concepts such as debugging, testing, and abstraction. Wing and her colleagues added some of the same concepts mentioned by NRC (2010) such as logic, algorithms, parallelism, pattern matching, recursion and procedural thinking to the earlier 2006 and 2008 formulations (Cuny et al., 2010).

Barr and Stephenson (2011) and The Computer Science Teachers Association (CSTA), as well as the International Society for Technology in Education (ISTE), were the first to frame CT domains in the specified context of grades K-12. Barr and Stephenson (2011) provided examples of CT applications in disciplines such as Math, Science, Social Studies, & Language Arts (see table 2). CSTA and ISTE developed a framework for grades K-12 that includes nine concepts of CT (CSTA & ISTE, 2011). They also provide a list of problem-solving characteristics that CT includes, such as "logically organizing and analyzing data" and dispositions/ attitudes that are essential to CT such as "persistence in working with difficult problems". Google (n.d.) reiterated the list of CT characteristics and dispositions/ attitudes put forth by CSTA and ISTE (2011). Google also adds a list of CT concepts/ mental processes (e.g. abstraction, algorithm design, decomposition, pattern recognition) and tangible outcomes (e.g. automation, data representation, pattern generalization) associated with solving problems in computing (Google, n.d.; Lodi, 2020).

As the body of CT literature grew, a plethora of review articles followed, some of which attempted to reach a consensus about the domains underlying CT. Selby and Woollard (2013)

reviewed some of the same CT domains as mentioned by previous scholars but excluded ones such as automation and modeling/ simulation from their definition because of lack of appearance within the previous CT literature of that time. Kalelioğlu et al. (2016) conducted an inductive qualitative content analysis on 125 CT papers. The authors analyzed CT definitions and features of CT domains. They then formulated their own framework for CT as a problem-solving process. Shute et al. (2017) described CT as having 6 main facets (see table 2). As shown in table 2, [www.CSunplugged.org](http://www.CSunplugged.org) categorized CT into six skills that are used throughout their lessons to teach unplugged CT to children (Bell & Lodi, 2019). The skills were based on previous work by Selby and Woollard (2013). Ezeamuzie and Leung (2021) classified CT components as either “significant components” or “non-significant components” and found that “abstraction” was the most mentioned subdomain in their literature review. Abstraction has also been mentioned as a central concept of CT by a plethora of other authors (Aho, 2012; Cetin & Dubinsky, 2017; Grover & Pea, 2013; ISTE & CSTA, 2011; Kalelioğlu et al., 2016; Wing, 2006, 2011). Similarly, Tikva and Tambouris (2021) reviewed 57 papers and listed the most common CT elements which were “abstraction”, “sequences” and “loops” among other domains.

Nearly all categorizations of CT domains draw upon the computer programming nomenclature, which furthers the perception that CT and coding are inexorably linked. Although educators have recognized that students can be introduced to the concepts embodied in the various CT domains through “unplugged” exercises, the CT domain nomenclature tends to be based on CS terms and coding-related concepts.

### ***Powerful Ideas of CT for Early Childhood***

Among the many CT domains that have been enumerated by authors in this field, a subset can be considered developmentally appropriate constructs for young children. A

developmentally appropriate construct is one that can be learned and understood at a particular stage of cognitive and social/emotional development. Developmental stage can impact a child's ability to understand certain CS concepts and their readiness for CT assessment (Chen et al., 2017). Bers (2018) used the various domain definitions described above to identify seven powerful ideas of CS and CT that could be considered developmentally appropriate for early elementary school children. These included the following:

1. **Algorithms:** A series of ordered steps that help solve problems or complete tasks (Yadav et al., 2017b). Sequencing is related to algorithmic thinking. Some have suggested that one exercises algorithmic thinking in everyday life activities such as brushing teeth or following a cooking recipe (Relkin & Strawhacker, 2021; Yadav et al., 2017b). Others suggest that sequencing and algorithmic thinking differ because algorithmic thinking must always be used to control “some abstract machine or computational model without requiring human judgment.” (Denning, 2017). Bers (2018) suggested that it is helpful to teach the concepts of algorithmic thinking and sequencing together to young children. As they get older, it becomes possible for them to untangle the two concepts.
2. **Modularity:** Modularity uses decomposed units (or modules) to break down complex processes and make them easier to handle. The modules can then be combined and/or reused to create a more complex process (Bers, 2018; Shute et al., 2017). Bers (2018) states that unplugged modularity and decomposition are often practiced in early childhood and gives an example of breaking down steps needed to plan a birthday party.
3. **Representation:** Representation involves understanding and applying symbol systems (Grover & Pea, 2013). Representation allows us to collect and interpret data.

Understanding that concepts can be represented with symbol systems is foundational to early childhood math, literacy, coding, and other disciplines (Bers, 2018).

4. **Debugging:** Debugging is the process of systematically finding and fixing errors (bugs) to solve a problem or complete a task. It is often an iterative process involving testing and modifying (Grover & Pea, 2013). It often involves using skills such as logical thinking, problem solving, evaluation, and perseverance (Bers, 2018). In early childhood, children learn to “debug” as part of their everyday lives, for example, by fixing mismatched socks or by identifying and editing mistakes in their writing.
5. **Control Structures:** Control structures involve a set of rules that only operate under certain conditions and determine the order or sequence of subsequent events in an automated process (Bers, 2018). As children develop abstract reasoning skills, they may engage with control structures involving repeat functions, loops, conditionals, events, and nested structures. Use of statements such as “if, then, else, or while” often indicate that conditionals contributing to control structures are being used (Relkin & Strawhacker, 2021).
6. **Hardware/Software:** Hardware and software can be seen as a basis for understanding automation. In early childhood, children begin to understand the differences between hardware and software as well as the relationship between the two. As children grow they start to recognize that smart objects are programmed by humans and are not magical (Bers, 2018).
7. **Design Process:** In computer science, creative design is an iterative process that involves using technological materials to create, design, and express oneself (Brennan & Resnick, 2012). Bers (2018) describes the design process in early childhood as a series of often

iterative steps that have no correct order or starting point: ask, imagine, plan, create, test and improve, and share.

Although the above definitions appear to describe discrete concepts, many of the domains of CT are closely related and may therefore be interdependent. Few formulations of CT have taken these interactions into account. For the purposes of this dissertation, domains will be treated as separate entities and interactions between domains will not be considered.

### **CT Assessment in Young Children**

Defining CT and its constituent domains has provided an important foundation for the creation of CT assessments. Designing a developmentally appropriate CT instrument for young children is challenging. For example, a kindergarten student may not be able to fully understand abstract CT/ programming concepts such as complex “if-then” conditionals (Barrouillet & Lecas, 1999; Janveau-Brennan & Markovits, 1999; Muller et al., 2001). Aspects of abstract representations such as programming variables may be inaccessible. They may express magical thinking as an explanation for the action of computers and other technology (Flavell et al., 1993; Mioduser et al., 2009). Cognitive development in early elementary school progresses rapidly, which may necessitate using different instruments in different grades (as is the case in the present study). These and other considerations must be taken into account when designing CT assessments for young children (Tang et al., 2020).

Initial attempts to create CT instruments for early age groups used techniques such as portfolio analysis, including interviews and/or observational methods (Bakala et al., 2021). For example, Mioduser and Levy (2010) presented pre-programmed LEGO robotics construction tasks to kindergarten-age children and their CT level was qualitatively assessed by analyzing the terms that children used to describe the robot’s actions as it navigated through a constructed

environment. In that study, children who attributed the robot's actions to magic were given low CT skills ratings, whereas those who provided mechanical explanations were considered more advanced. Wang et al. (2014) used a similar approach with five to nine year-old-children, who were asked open-ended questions about a tangible programming task that they created called "T-maze". "T-maze" uses TopCode to convert physical programs into digital code (Horn, 2012). The researchers identified elements of CT in the children's responses (e.g., abstraction, decomposition) as a basis for determining whether the children grasped these concepts. Bers et al. (2014) created a checklist to assess programs created by kindergarteners (ages 4.9 to 6.5 years old) exposed to a tangible and graphical programming language called CHERP. During one session, children were tasked with programming their robot to dance the "Hokey Pokey". The researchers then assessed four CT concepts by scoring children's projects on a Likert-type response scale. Moore et al. (2020) used task and interview techniques to assess CT. Three participants were videotaped while they were interviewed and performed tasks using the Code and Go Robot Mouse Coding Activity (Learning Resources, Vernon Hills, IL). Researchers explored qualitatively how children use representations and translations to invent strategies for solving problems. Portelance and Bers (2015) conducted an exploratory study that assessed CT in young children by analyzing ScratchJr artifact-based video interviews of students in pairs. Researchers then analyzed videos of the dyads using holistic coding to identify categories.

Although interview and observational assessments can provide useful information about a child's CT skills, the format of these assessments and the time they require makes them unsuitable for administration outside of specific research settings. The interview-based approach is time-consuming, subjective, and may be further limited by a child's capacity to verbalize their thought processes.

Some recent attention has been paid to creating activity-based CT assessments for young children. Marinus et al. (2018) created the Coding Development (CODE) Test 3–6 (for children between three and six years of age), which uses the robot Cubetto. CODE requires children to program the robot to go to a specified location on a mat by inserting wooden blocks into a “remote control.” The task is to either build the program from scratch or debug an existing program. Children are given maximally three trials to complete each of the 13 items, with more points being awarded if fewer attempts are needed. Although the authors state that CODE is meant to measure CT, their assessment requires coding knowledge which raises the possibility that their assessment conflates coding with CT skills.

### **Unplugged CT Assessment**

It is advantageous to be able to measure CT skills in children regardless of whether they have past knowledge or experience with computer programming (Grover et al., 2014). With this in mind, my colleagues and I at the DevTech Research Group at Tufts University and other groups began exploring the use of code-free instruments to assess CT skills in children. In recent years, unplugged challenges have started to be used for the purposes of assessment in older children (e.g., Dagienė & Stupurienė, 2016; Román-González et al., 2018). Unplugged assessments offer certain advantages because they do not rely on a particular computer language or curriculum and may therefore be “purer” reflections of CT abilities (Dagienė, & Futschek, 2008).

Since 2018, at least five different studies characterizing unplugged CT assessments designed specifically for young children have been published. The measures include: the Computational Thinking Assessment (CTA) (Tran, 2019), *TechCheck* (Relkin et al., 2020; Relkin & Bers, 2021), The Computational Thinking Test for Beginners (BCTt) (Zapata-Cáceres et al.,

2020), the Computerized Adaptive Programming Concepts Test (CAPCT) (Hogenboom et al., 2021), The Competent Computational Thinking Test (cCTt) (El-Hamamsy et al., 2022). All five use unplugged challenges to probe CT domains and can be administered to children who lack prior coding experience. These instruments differ in the types of unplugged challenges they include, the CT domains assessed, the age ranges they cover, and the time required to complete and score the respective assessments (see Table 3). The cCTt is an updated, more difficult version of the BCCt designed specifically for 3rd and 4th grade students. Some of the concepts probed by the cCTt, the BCCt, the CAPCT, and the CTA such as complex conditionals may be too difficult for very young children on developmental grounds (Barrouillet & Lecas, 1999; Janveau-Brennan & Markovits, 1999; Muller et al., 2001). In addition, the CAPCT and the CTA require more advanced language and mathematical skills than typical K-2 students possess.

**Table 3**

*A Comparison of Five Unplugged CT Measures for Young Children (ages 2-10)*

	<b>The CTt for Beginners (BCTt)</b>	<b>The Competent Computational Thinking Test (cCTt)</b>	<b><i>TechCheck</i></b>	<b>Computerized Adaptive Programming Concepts Test (CAPCT)</b>	<b>Computational Thinking Assessment (CTA)</b>
<b>CT Concepts</b>	Sequences, Loops (Simple, Nested), Conditionals (If-Then, If-Then-Else, While)	Sequences, Loops (Simple, Nested), Conditionals (If-Then, If-Then-Else, While)	Algorithms, Modularity, Debugging, Hardware/Software, Control Structures, Representation	Basic Sequences, Loops, Conditions (If & If-Else Statements), Debugging, Multiple Agents, Procedures, Generalization	Sequences, Algorithms, Loops, Debugging, Conditionals
<b>Format Type</b>	Pen and paper Multiple choice	Pen and paper Multiple choice	Pen and paper Online Multiple choice	Online Adaptive	Pen and paper Yes/No Prose responses
<b>Items</b>	25 items	25 items	15 items	4486 items (utilizes alternative forms of the same items)	10 items



<b>Administrator Needed</b>	Yes	Yes	Yes	No	No
<b>Average Testing Time</b>	40 minutes	30-35 minutes	13 minutes	Children play for as long as they want	6-10 minutes
<b>Sample</b>	299 students	1519 students	1844 students	93,341 students	183 students
<b>Age Range</b>	5-12 (1- 6 <sup>th</sup> grade)	7-9 (3-4 <sup>th</sup> grade)	2-9 (PreK – 2 <sup>nd</sup> grade)	6-13 (1– 7 <sup>th</sup> grade)	N/A (3 <sup>rd</sup> grade)

---

*Note.* BCTt: The CTt for Beginners (Zapata-Cáceres et al., 2020); cCTt: the Competent Computational Thinking Test (El-Hamamsy et al., 2022); *TechCheck* (Relkin et al., 2020), CAPCT: the Computerized Adaptive Programming Concepts Test (CAPCT) (Hogenboom et al., 2021); CTA: the Computational Thinking Assessment (Tran, 2018).

### **The *TechCheck* CT Assessment**

I have gone into considerable detail in describing unplugged CT assessment because the current study leverages the benefits of unplugged assessment for evaluating CT skills of coding-naive children, which could not be readily done with instruments that require platform-dependent coding skills.

This study employs *TechCheck* (Relkin et al., 2020), which was developed based on six of the seven Powerful Ideas of CS put forth by Bers (2018). Design Process, one of the Powerful Ideas, was excluded because it is an iterative and open-ended process that does not lend itself to a short, multiple-choice assessment. A group of nineteen evaluators (CS researchers, educators and students) with various levels of expertise in CT were assembled to establish a consensus about which of the domains of Bers' Seven Powerful Ideas were embodied in each of the questions. Inter-rater agreement was then assessed. There was an average agreement of 81% among raters. Fleiss' Kappa indicated consensus among evaluators about the CT domain most associated with each question  $\kappa = 0.63$  (95% CI)  $p < 0.001$ . Although all prototypes were judged to probe the intended CT domains, some questions were rejected because their content was

judged to fall outside the common knowledge base of typical five- to nine-year-old children (Relkin et al., 2020).

*TechCheck* was initially tested in a cohort of 768 first and second graders (ages 5-9) participating in a research study involving the CAL-KIBO curriculum. Students from eight schools were involved in the coding, robotics, and literacy curriculum for 2 hours per week over 6 weeks (second graders) or 7 weeks (first graders). Scores on *TechCheck* showed good reliability and validity according to classical test theory (CTT) and item response theory (IRT), models that are commonly used to better understand the relationship of assessment items to the underlying concepts being measured (Kingsbury & Weiss, 1983). The mean difficulty index of all items was  $-1.25$  (range =  $-2.63, .7$ ), the mean discrimination index was  $1.03$  (range =  $0.65, 1.41$ ). The coefficient alpha indicated a moderate level of internal consistency ( $\alpha = 0.68$ ) (Hinton et al., 2004). The assessment scores were normally distributed, and the assessment readily distinguished among young children with different CT abilities. *TechCheck* scores correlated moderately ( $r = .53, p < .001$ ) with the previously validated CT assessment tool called TACTIC-KIBO that classifies each child in one of four programming proficiency levels derived from the Developmental Model of Coding (Vizner, 2017). Scores were also highly correlated with expert ratings of children's CT skills, indicating criterion validity.

In a prospective longitudinal study, Relkin et al. (2021) used *TechCheck* to compare children receiving the CAL-KIBO programming curriculum ( $N = 667$ ) to a control group ( $N = 181$ ) who participated in typical classroom activities without coding (No-CAL). Over the course of the study, children who received CAL-KIBO improved on *TechCheck* ( $M_{\text{change}} = 0.94, p < 0.001$ ) whereas the No-CAL group did not change significantly ( $M_{\text{change}} = 0.27, p = .07$ ). The mean difference in baseline scores (2.54 points) divided by the mean difference in the ages of first and

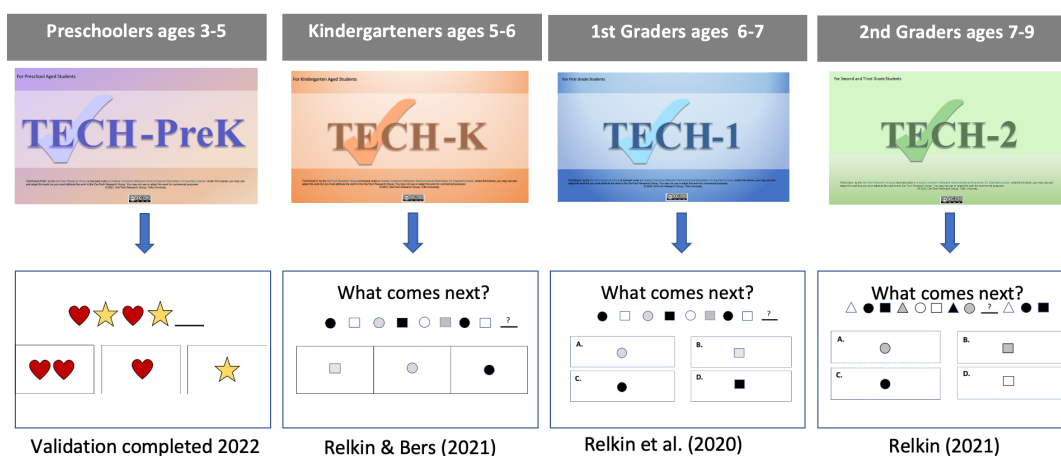
second graders (1.30 years) was used to calculate the approximate expected change in *TechCheck* scores between first and second grade (1.95 points/year). The CAL-KIBO group's *TechCheck* change scores equated to the change in scores estimated to occur over approximately 6 months of typical development. Generalized Linear Mixed Model (GLMM) and Bayesian linear mixed modeling revealed that exposure to the CAL-KIBO curriculum predicted the *TechCheck* outcome score, considering differences in baseline *TechCheck* performance and other demographic and environmental effects. Regression analyses indicated that baseline *TechCheck* scores predicted performance on KIBO Mastery Challenges (KMCs), a formative measure of coding proficiency (Relkin & Bers, 2020). In addition, teacher proficiency in CT as measured by the TACTIC-KIBO instrument prior to implementation of the CAL-KIBO curriculum as well as school significantly predicted changes in student learning as measured by *TechCheck* (Relkin & Bers, 2021). Children who received CAL-KIBO showed the most improvement in the CT domains algorithms, modularity, and representation.

The results observed with *TechCheck* scores in the above-described longitudinal study indicate that the assessment is sensitive to change. The longitudinal study was the first to use an unplugged CT assessment designed for young children with a sizable control group of coding-naïve students. Although results using *TechCheck* in first and second graders were encouraging, there was a noticeable ceiling effect in the second-grade cohort manifesting in a smaller window to observe change compared to the first-grade cohort. The observed ceiling effect motivated the creation of grade-specific versions of *TechCheck* (Figure 2). The original version was renamed "*TechCheck-1*" for first graders. I created and collected validity evidence for *TechCheck-K* for Kindergarten aged children (5-6 years old) (Relkin & Bers, 2021) and *TechCheck-2* for children in second grade (7-9 years old) (Relkin, 2021). *TechCheck-Pre-K* was

created more recently and validated for preschool children between three to five years of age (Relkin & Bers, under preparation). In the preschool version, item difficulty was reduced and all text was removed to limit distraction. *TechCheck-PreK* requires a proctor to read a script to the child and currently must be administered in a one-on-one setting.

## Figure 2

*Example Algorithms items from the grade-specific versions of TechCheck*



## Method

This dissertation draws on data obtained with the four grade-specific versions of *TechCheck* to explore how young children learn and develop CT skills.

### Experimental Design

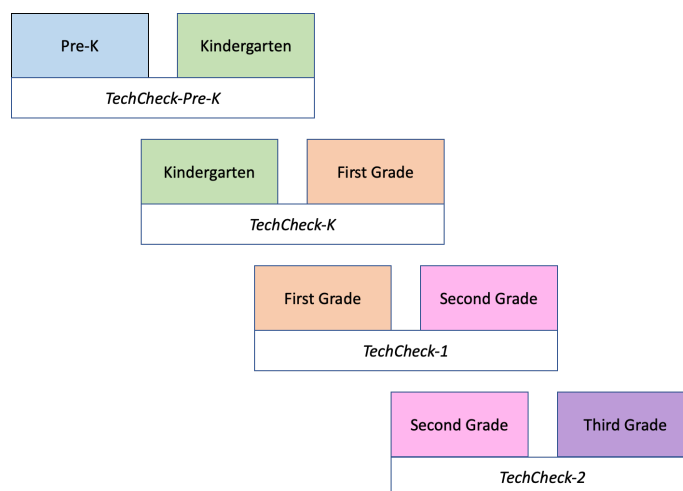
**RQ1: To what extent do CT skills differ in coding-naive children in pre-K through third grade?**

The strategy for addressing this question involves the analysis of *TechCheck* data collected from coding-naive children in grade pairs, as shown in Figure 3. I collected data from coding-naive children receiving the version of *TechCheck* designed for their grade and compared their performance to that of coding-naive students in one grade higher who were administered the same *TechCheck* version. The data analyses include the Welch two sample *t*-tests to examine

statistical significance of differences between means, linear regression to estimate the annualized rate of change between grades, and Crossed Random Effects Modeling for domain analysis examining the percent correct responses in each of the six CT domains for children in each grade pair.

### Figure 3

*Grade pairings used to address RQ 1.*



### **RQ2: What demographic and environmental factors predict CT skills in coding-naive children?**

To answer this question, I conducted Linear Mixed Modeling (LMM) on *TechCheck* data for coding naive children in grades Pre-K, K, 1 and 2 who received the grade-optimal version of *TechCheck*. The outcome variable for the model is *TechCheck* total score. Fixed effects included age, grade, gender, disability status and race/ethnicity. Random effects included Teacher/Classroom, School and District. Possible interactions were also included in the model. I previously created a normalized scoring system to compare *TechCheck* total scores across grades using the grade-specific versions of *TechCheck* (Relkin et al., 2022; Relkin, 2022). The

normalization uses z-scores to take into account differences in baseline score distributions for the grade-specific versions of *TechCheck*. Percentile ranks were then derived from the z-scores.

**RQ3: How do coding educational interventions (specifically CAL-ScratchJr, CAL-KIBO, and codeSpark) affect the rate of acquisition of CT skills overall and CT domains?**

To explore this question, I examined longitudinal outcomes from three coding interventions in which *TechCheck* was administered before and after children participated either in a curriculum teaching them to code or regular classroom activities (control group). Analysis methods included paired sample *t*-tests and Cohen's *d* effect size calculations, as well as LMMs of longitudinal changes in total score and domain change scores.

**Data Sources**

The analyses described in this dissertation include newly collected data (Table 4, study 1) and data from past studies. Newly collected data included students in preschool, kindergarten and first grade with no formal coding education prior to enrollment in the study. The historical data used for this dissertation (Table 4, studies 2-8) included students in pre-kindergarten through 3<sup>rd</sup> grade who were enrolled in studies using *TechCheck* conducted at various times between 2019 and 2022. The children were between the ages of two and nine and enrolled in schools located in Virginia, California, Minnesota, Arkansas, Massachusetts, and Missouri.

All studies were approved by the Tufts Social and Behavioral Research IRB (protocols 1105050, 1810044) and/or the University of California Los Angeles IRB (protocol 19-002034). In addition to parent/guardian consent, student verbal assent was obtained prior to the administration of *TechCheck*.

**Table 4***Summary of Data Sources*

	Study	<i>N</i>	RQ	Location	Year	Control Group	Grades Assessed
1	Dissertation Data Collection	72	1,2	NYC; MA	2022		K,1
2	CAL-KIBO	848	1,2,3	VA	2019-20	X	K,1,2
3	Pilot CAL-ScratchJr	161	1,2,3	MN; AR; CA	2020-21		K,1,2
4	DOE Impact CAL-ScratchJr	1244	1,2	MA; RI	2021-Ongoing	X	K,1,2
5	UCLA CRESST codeSpark Academy	72	1,2,3	CA	2021-22		1,3
6	CAL-ScratchJr Camp	59	1,2	Virtual	2020		K,1,2
7	Head Start	79	1,2	MO	2021-22	X	PreK
8	Horizons CAL-KIBO	55	1,2	MA	2021-22		PreK

**Coding Interventions**

The platforms and curricula that were employed as educational interventions in these studies are as follows:

***KIBO***

KIBO is a screen-free robotics kit that is developmentally appropriate for PreK through second grade children. KIBO teaches children to code using tangible wooden blocks that can be sequenced and scanned with the robot's embedded barcode reader. Each block represents an action that the robot performs. In addition to the programming blocks, children can connect sensors, modules, and art platforms to their robot which gives them a unique opportunity to use their creativity to make personally meaningful projects.

### ***ScratchJr***

ScratchJr is a free screen-based programming application for young children grades K through 2. ScratchJr allows children to engage with open-ended programming by utilizing virtual coding blocks. Children can create their own games, stories and other projects. ScratchJr allows children to learn programming concepts such as sequencing, loops, conditionals, parallelism, and more. Children can also exercise their creativity by customizing their projects with backgrounds and characters. As of October 2021, ScratchJr had been downloaded over 27 million times across 194 different countries (Bers et al., 2022).

### ***codeSpark***

codeSpark Academy is a screen-based programming application for children ages 5-10. The self-paced game uses virtual programming blocks to engage children in multiple levels that vary in terms of puzzle difficulty. The game covers various coding concepts such as sequences, parameters, loops, events, and conditionals. In addition to solving pre-determined puzzles, codeSpark includes a more open-ended “Game Maker” in which children can create and customize their own puzzles that they can then play themselves or share with others.

### **Prior Research on Included Coding Platforms**

Both KIBO and ScratchJr have been subjected to numerous studies that have found benefits for young children including, but not limited to, promoting self-expression and social-emotional skills, sequencing skills, problem solving, parent-child interactions, communication, coding, and CT skills (Bers, 2020; Govind et al., 2020; Strawhacker & Bers, 2019; Strawhacker et al., 2017; Relkin et al., 2020; Relkin et al., 2021)

To date, limited research exists on the codeSpark platform. One exploratory study conducted by KnowProgress with 27 children found that children’s sequencing, CT, and



problem-solving skills all significantly improved after just 90 minutes playing codeSpark (Denniston et al., 2015). In another study conducted by WestEd, 93 children either assigned to receive codeSpark or a control group were administered a sequencing assessment. Although children's scores increased from baseline to after the intervention, there was no significant difference between treatment and control (Grillo-Hill et al., 2019). These pilot studies were preliminary and more research is needed to determine codeSpark's impact.

## **Coding Curricula**

### **Coding as Another Language (CAL) Curricula**

The CAL curricula are designed to teach coding and CT to young children while simultaneously promoting literacy skills (Bers, 2019; Hassenfeld et al., 2020). The curricula promote self-expression analogous to that provided by use of symbolic written languages (Bers, 2018, 2019). Bers' Powerful Ideas provided the main content foundation for the curricula. Beyond coding, CT, and literacy, these curricula strongly emphasize the building of social-emotional skills. The curriculum aligns with the Positive Technological Development (PTD) framework, which encourages students to engage in six positive behaviors: content creation, creativity, choices of conduct, communication, collaboration and community building (Bers, 2008, 2012, 2017, 2020). Each lesson in the curriculum also promotes multiple virtues from "The Pallet of Virtues": Curiosity, Perseverance, Open-Mindedness, Optimism, Honesty, Patience, Generosity, Gratitude, Forgiveness, and Fairness (Bers, 2022).

The CAL curricula are platform-specific. There are three curricula for the ScratchJr platform (CAL-ScratchJr) designed for kindergarten, first and second grade respectively called CAL-ScratchJr. There are four curricula for the KIBO platform (CAL-KIBO) designed for preschool, kindergarten, first, and second grade students. The curricula align with various

standards for CS, literacy, and math such as the Common Core English Language Arts (ELA)/Literacy Framework, Common Core Math Standards, ISTE Student Standards, and the K-12 Computer Science Framework.

### **codeSpark Academy Curriculum**

The codeSpark curriculum consists of twenty-five 30-minute lessons that cover ten CS concepts. The lesson activities are categorized in an “Engage, Explore, Enrich” framework. “Engage” involves children first being introduced to the concept, typically through unplugged activities. “Explore” involves hands-on codeSpark game-play. Lastly, “Enrich” involves connecting the concept to the real world, often through additional unplugged activities and/or discussion. The curriculum aligns with math, science, and computer science standards such as CSTA (2017) and the Common Core English Language Arts (ELA)/Literacy Framework.

### **Comparison of Curricula**

The interventions described above have many similarities and differences. There is only one curriculum for codeSpark that covers all elementary school ages, whereas CAL-KIBO and CAL-ScratchJr feature 3-4 different curricula based on a child’s grade. CAL-KIBO and CAL-ScratchJr curricula were both designed to cover the same topics of coding and literacy skills and to promote social-emotional development. However, the coding platforms they involve (screen-free robot vs. screen-based interface) may afford the child inherently different experiences.

The coding and CS concepts in which the children engage when participating in the CAL-KIBO, CAL-ScratchJr and codeSpark curricula are very similar. For example, they all involve children debugging, engaging in the design process, learning about loops, and engaging in algorithmic thinking. In general, throughout the CAL curricula, children first have hands-on

experiences where they freely explore concepts and then later their learning is reinforced through direct teaching (as per constructionism pedagogy). This pedagogical method is reversed in the codeSpark curricula as concepts and terms are typically first explicitly taught then later explored by children (as recommended by Wing, 2008). Both curricula involve ill-structured and well-structured activities. However, CAL largely emphasizes open-ended problem solving and codeSpark largely involves puzzles with specific solutions. Both curricula suggest opportunities for differentiation in which teachers can modify the content to address the specific needs of their students.

### **CT Assessment Administration**

Due to the COVID-19 pandemic and school resource limitations, five formats of administration of the various grade-specific versions of the *TechCheck* assessment were implemented across the various studies contributing to this dissertation. Regardless of the format, proctors were trained to administer the assessment in a consistent fashion. Students were administered the assessment in one of the five formats shown in Table 5. Across all formats of administration, each question was read out loud to the students by a proctor who asked them to provide a single answer from a set of multiple choice responses. There were two practice questions that were included at the beginning of the assessment to ensure that children felt comfortable with the format of administration and knew how to indicate their answers. Students were allowed to take breaks for up to 5 minutes during the assessment. Students were instructed to guess if they did not know the answer.

#### ***In-person via tablet group administration***

For this type of administration, each student received a tablet with a link to a secure online survey platform with the *TechCheck* assessment on the home screen. Proctors projected a

PDF of the assessment onto a board in front of the class and read each question out loud to the group. This enabled pre-literate children to easily follow along. Students were instructed to tap the correct answer. The assessment was forced-choice and the survey could not progress until an answer had been inputted.

### ***In-person via tablet one-on-one administration***

In this format, students had one-on-one sessions with a proctor in which *TechCheck* was administered on tablets. The format of the *TechCheck-PreK* assessment involves using a script (instead of written text on the questions) and proctors allowed children to either tap the correct answer or verbally indicate their answer. The assessment was forced-choice and the survey could not progress until an answer had been inputted.

### ***In-person via paper and pencil***

Pen and pencil administration involved giving students a packet containing the assessment printed in color. Proctors also projected a PDF of the assessment onto a board in front of the class so that preliterate children could easily follow along. After proctors read each question, students circled their chosen answer. Proctors checked the packets throughout the session to ensure each question was answered by the students.

### ***Virtually via one-on-one Zoom sessions***

Virtual administration was employed when students were either located at home or at school and required the use of private Zoom sessions owing to the COVID-19 pandemic. The proctors shared their screens and navigated to the *TechCheck* secure online survey platform where they read each question out loud and asked children to verbally indicate their responses. Proctors then noted the students' responses on the secure online survey platform. The assessment was forced-choice and the survey could not progress until an answer had been inputted.

### ***Virtually via Zoom group administration***

Some students were located at home due to the COVID-19 pandemic and joined a group link to a Zoom session with their peers and a proctor. The proctor administered the assessment through an online platform that allowed the proctor to control the advancement of slides and all children to see each question at the same time. After the proctor read the questions, students clicked on their answers. For this particular administration format, the automation did not mandate a response to every question. The proctors continued administration after the majority of students had inputted their answers.

**Table 5**

*Administration Format of TechCheck used in each study*

	Study	Format
1	Dissertation Data Collection	Virtually via one-on-one Zoom sessions; In-person via tablet one-on-one administration
2	CAL-KIBO	In-person via tablet group administration
3	Pilot CAL-ScratchJr	Virtually via one-on-one Zoom sessions
4	DOE Impact CAL-ScratchJr	Virtually via one-on-one Zoom sessions
5	UCLA CRESST codeSpark Academy	Virtually via Zoom group administration; In-person via paper and pencil
6	CAL-ScratchJr Camp	Virtually via one-on-one Zoom sessions
7	Head Start	In-person via tablet one-on-one administration
8	Horizons CAL-KIBO	In-person via tablet one-on-one administration

### **Data Analysis**

#### **Calculation of CT Domain Scores**

To assess performance within the six powerful ideas probed by *TechCheck*, domain score analysis was carried out. To perform the analysis, I took clusters of questions that were

associated with individual CT domains and calculated the average percentage of correct answers in that domain. There were 2 to 5 items per CT domain with the questions being determined by the *TechCheck* grade-specific version employed (see table 6). Results were averaged over all of the questions in the domain and expressed as a percentage. Each student was given an average percentage correct score for each of the six domains, which was used to prepare bar charts and build Crossed-Random Effect models. Table 6 shows which questions probe each of the six CT domains for the various *TechCheck* versions.

**Table 6**

*Mapping of Questions on Versions of TechCheck by CT domains the Questions Probe*

CT Domain	<i>TechCheck-PreK</i>	<i>TechCheck-K</i>	<i>TechCheck-1</i>	<i>TechCheck-2</i>
Hardware/Software	Q1, Q2	Q1, Q2	Q1, Q2	Q1, Q2
Debugging	Q3,Q4	Q3,Q4	Q3,Q4	Q3,Q4
Modularity	Q8,Q9	Q6,Q7	Q6,Q7	Q5,Q6, Q7
Algorithms	Q5, Q6, Q7, Q12, Q13	Q5, Q8,Q9, Q10, Q11	Q5, Q8,Q9, Q10, Q11	Q8,Q9,Q10, Q11
Representation	Q14,Q15	Q12,Q13	Q12,Q13	Q12,Q13
Control Structures	Q10,Q11	Q14,Q15	Q14,Q15	Q14,Q15

### Statistical Analyses

All statistical analyses were conducted in R (Version 2021.09.2) using R Studio version 4.1.2 (R Core Team, 2021). All plots were generated using the “ggplot2” function in R (Wickham, 2016). I built Linear Mixed Models (LMMs) and Crossed Random Effects Models using the “lmer” function in R Studio (Bates et al., 2015). For the majority of analyses, LMMs were chosen over regression due to the nested nature of the data (e.g., students within classrooms within schools).LMMs are multilevel models (assuming linearity of relationship between

predictors and the outcome) that contain both fixed and random effects. Fixed effects are variables that are expected to predict the outcome variable. Random effects are categorical and typically grouping variables that are ideally accounted for, such as student or teacher. There can be random effects of slopes or intercepts. Crossed Random Effect Models can be conducted when every observation at level 1 is nested within two random variables at level 2 (in this case, domain scores were nested within domains and students). Missing cases were removed with list-wise deletion for the purposes of this study. Pre-analysis data screening prior to each model showed adequate normality of the variables used in all of the models.

In each case, I started with an empty model including just the outcome variable and a random effect of an intercept. Then I added each predictor to the model sequentially and conducted likelihood ratio tests to examine whether those predictors improved the model fit. I then added interaction terms. If the model fit did not improve with the addition of that predictor or interaction, it was removed from the final model. Post analysis examinations of residuals were conducted on the final model to ensure normality based on histogram and P-P plots. VIF ( $<10$ ) and Tolerance ( $>.02$ ) values were used to evaluate multicollinearity, and I also tested the assumptions of homoscedasticity of residuals and influential outliers (Field, 2009).

## **Results**

Results obtained for the three research questions are presented below. Each section consists of a description of the demographics of the study participants followed by the results for that question.

**RQ 1: To what extent do CT skills differ in coding-naive children in preschool through third grade?**

*RQ 1 Part 1: Examining differences in mean TechCheck scores across grade pairs*

## Participants (RQ1 Part 1)

Data from a total of 2775 coding-naive students were obtained from eight different studies in which students were administered a version of *TechCheck* that was either optimal for their grade or designed for one grade below. The data reflect one administration of *TechCheck* per child carried out prior to their engagement in any formal coding instruction. Altogether, 2121 children were given the assessment designed for their grade and 654 students were administered the version of *TechCheck* designed for one grade below their own. Children were between the ages of 2.81 and 9.04 years. The average age difference between the grade pairs was 1.28 years for *TechCheck-PreK*, 0.64 years for *TechCheck-K*, 1.12 years for *TechCheck-1*, and 0.25 years for *TechCheck-2*. Sample sizes ranged from a minimum of 17 students in Grade 1 taking *TechCheck-K* to a maximum of 935 students in first grade taking *TechCheck-1*. The third grade sample had a large percentage of students that were English Language Learners (52.78%). Table 7 shows the demographics for each grade included in this analysis.

I first examined differences in *TechCheck* scores for each of the four grade pairs receiving the grade-specific versions of *TechCheck*. Figure 4 shows the resulting density plots of the four grade pairs. Descriptive statistics for the grade pair analyses are shown in Table 8. Because scores were relatively normally distributed, means were used to represent the central tendencies of each distribution. A substantial ceiling effect is evident in the kindergarten students taking the pre-K version of *TechCheck*. A smaller ceiling effect is present in the second grade cohort who were administered the first grade version. In the context of the current analysis, ceiling effects may lead to an under-estimation of the difference between the grade pairs.



**Table 7***Research Question 1 Demographics by Grade*

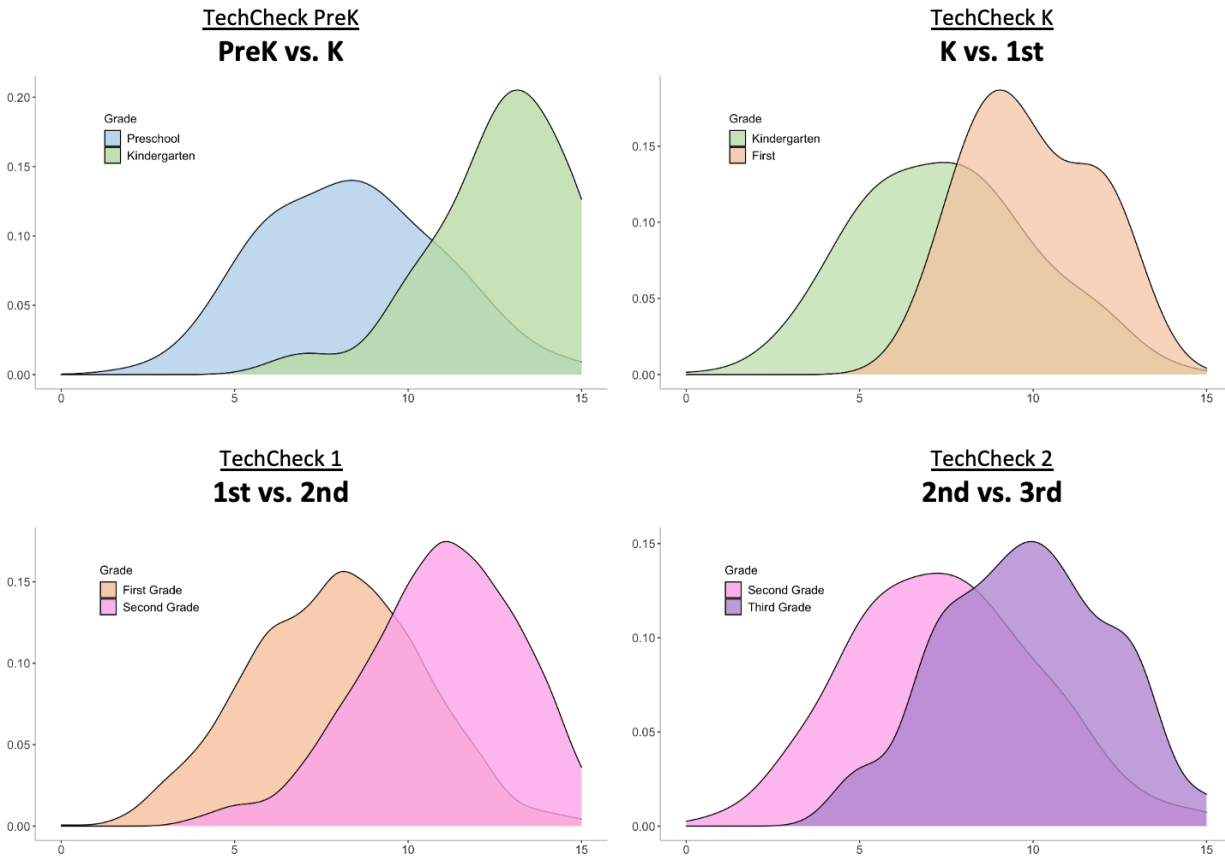
TC Version	TC -PreK		TC-K		TC- 1		TC-2	
Grade Pair	PreK	K	K	1	1	2	2	3
Number of students	173	27	395	17	935	574	618	36
Mean Age (SD)	4.02 (0.62)	5.30 (0.61)	5.86 (0.42)	6.50 (0.52)	6.50 (0.56)	7.62 (0.47)	7.81 (0.35)	8.06 (0.24)
Missing data	2	0	55	1	185	0	299	1
Gender								
Male	82	17	164	6	399	275	162	17
Female	84	10	171	9	400	295	157	19
Non-Binary	0	0	0	1	0	0	0	0
Missing data	7	0	60	0	136	4	299	0
Race								
Black/African American	96	1	34	0	154	262	20	0
Hispanic/ Latino	29	2	42	1	113	66	56	36
Biracial/Multiracial	7	0	23	0	42	40	15	0
White	23	12	220	5	398	184	206	0
Asian	7	9	12	10	30	13	16	0
Other	3	2	4	0	8	1	6	0
Missing data	8	1	60	3	190	8	299	0
Disability	7	0	38	0	77	57	33	0
Missing data	93	27	70	17	244	107	299	36

*Note. TC is the abbreviation for TechCheck*

The difference in *TechCheck* mean scores for the four grade pairs examined as well as the associated Welch two sample *t*-tests results are shown in Table 9. The *t*-tests show that there were significant differences in baseline *TechCheck* scores for all grade pairs.

**Figure 4**

*Density plot distributions by TechCheck grade pair*



**Table 8***Descriptive Statistics for paired grade analyses*

Grade	TechCheck Version	N	Mean	SD	Median	Min	Max
Pre-K	Pre-K	173	8.40	2.49	8	3	15
K	Pre-K	27	12.67	1.92	13	7	15
K	K	395	7.48	2.52	7	0	14
1	K	17	9.94	1.78	10	7	13
1	1	935	7.98	2.46	8	0	15
2	1	574	10.89	2.25	11	4	15
2	2	618	7.48	2.75	7	0	15
3	2	36	9.81	2.40	10	5	15

Plots of results of grade pair differences by grade and by grade adjusted for age are shown in Figure 5. Data on the ages of children in the respective grades were used to adjust for differences in the mean age of children in each grade pair. This exploratory analysis suggests that CT skills improve in students from PreK to grade 3 even in the absence of formal CS education.

**Table 9***TechCheck baseline grade pair results*

<b>TechCheck Version Administered</b>	<b>Grade Pair</b>	<b>ΔMean (SD)</b>	<b>T-value</b>	<b>Significance</b>
<i>TechCheck-PreK</i>	PreK-K	4.27	$t(703.81) = 15.92$	$p < .0001$
<i>TechCheck-K</i>	K-1	2.46	$t(18.86) = 5.46$	$p < .0001$
<i>TechCheck-1</i>	1-2	2.91	$t(1293.2) = 23.51$	$p < .0001$
<i>TechCheck-2</i>	2-3	2.33	$t(40.53) = 5.60$	$p < .0001$

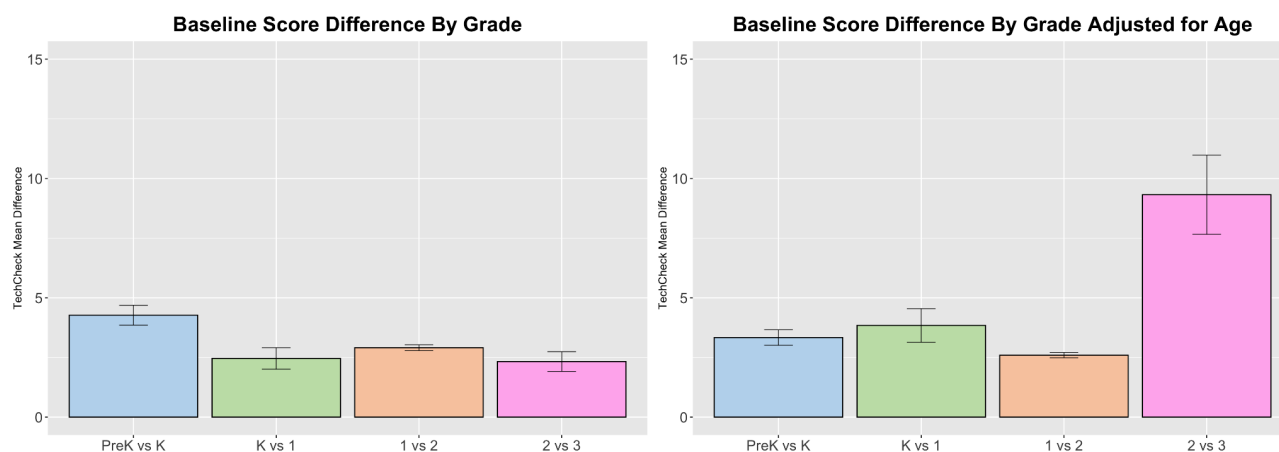
*Note.* This table shows that there is a significantly higher score across each grade pair of

*TechCheck*

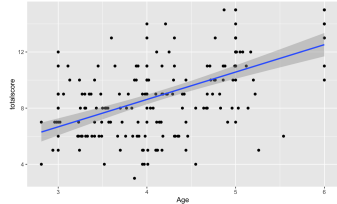
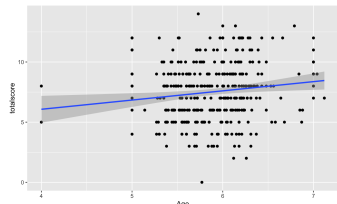
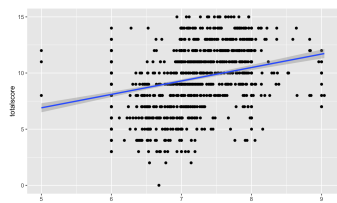
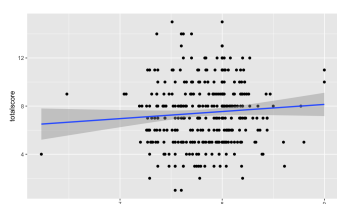
Next, I examined the differences based on age within each grade pair by conducting linear regression and calculating the coefficients for predicting *TechCheck* total scores as a function of age. Table 10 shows the predicted differences in *TechCheck* scores for the respective grade pairs as a function of age. Age was a significant predictor for preschool/kindergarten ( $\beta = 0.54$ ), kindergarten/ first grade ( $\beta = 0.13$ ), and first grade/second grade ( $\beta = 0.33$ ). Between preschool and kindergarten, 1.95 percent of the variance in baseline *TechCheck* score was attributable to the child's age  $F(1, 196) = 83.2, p < .001$ . Between kindergarten and first grade 1.91 percent of the variance in baseline *TechCheck* score was attributable to the child's age  $F(1, 354) = 6.91, p = .008$ . Between first and second grade, 10.84 percent of the variance in baseline *TechCheck* score was attributable to the child's age  $F(1, 1300) = 158.1, p < .0001$ . Age was not a significant predictor of total score between second and third grade ( $\beta = 0.07$ ),  $F(1, 353) = 2.14, p = .14$ . The lack of a predicted difference may relate to the small sample size of third graders as well as the small difference in age between the second and third graders (0.26 years).

### Figure 5

*Unadjusted and adjusted mean differences for grade pairs receiving TechCheck*



**Table 10***Predicted TechCheck Change Scores for grade pairs*

<b>Grade Pair</b>	<b>Predicted change per year</b>	<b>Standard Error</b>	<b>R<sup>2</sup></b>	<b>Significance</b>	<b>Relationship</b>
PreK-K	0.81	0.21	.29	$p < .001$	
K-1	0.77	0.29	.02	$p = .008$	
1-2	1.19	0.10	.11	$p < .001$	
2-3	0.59	0.40	0	$p = .14$	

**RQ 1 Part 2 Examining differences in TechCheck outcome within and across CT domains****Participants (RQ1 Part 2)**

Analysis of CT domain differences in coding-naive students was carried out in a subset of  $n = 2121$  from the same cohort of participants from preschool through second grade. This subset only included students who received the version of *TechCheck* designed for their grade. Students

one grade higher were not included in this analysis. The subset was composed of coding-naive students between the ages of 2.81-9.04 years who received *TechCheck* at one time point prior to any formal coding instruction. The sample consisted of  $n = 812$  (50.15%) female students and  $n = 807$  (49.84%) male students. The majority of students,  $n = 847$  (54.16%), were White followed in frequency by Black,  $n = 304$  (19.44%), Hispanic,  $n = 240$  (15.34%), Biracial or multiracial,  $n = 87$ , (5.56%), Asian,  $n = 65$  (4.15%), and “Other”  $n = 21$  (1.34%). The group characterized as “other” consisted of children identified as American Indian, Alaskan Native, Pacific Islander or Native Hawaiian. One hundred fifty five children were diagnosed with a disability such as autism, developmental delays, specific learning disabilities, specific language impairments, and other health impairments. For the purposes of the present study, children diagnosed with any disability were grouped together. The gender of  $n = 502$  students, the age of  $n = 541$ , the race/ethnicity of  $n = 557$  students, and the disability status of  $n = 706$  could not be determined because the data were either derived from pilot studies that did not collect those data or from ongoing studies in which demographic information was not yet available. Demographics for this sample are shown in Table 11.

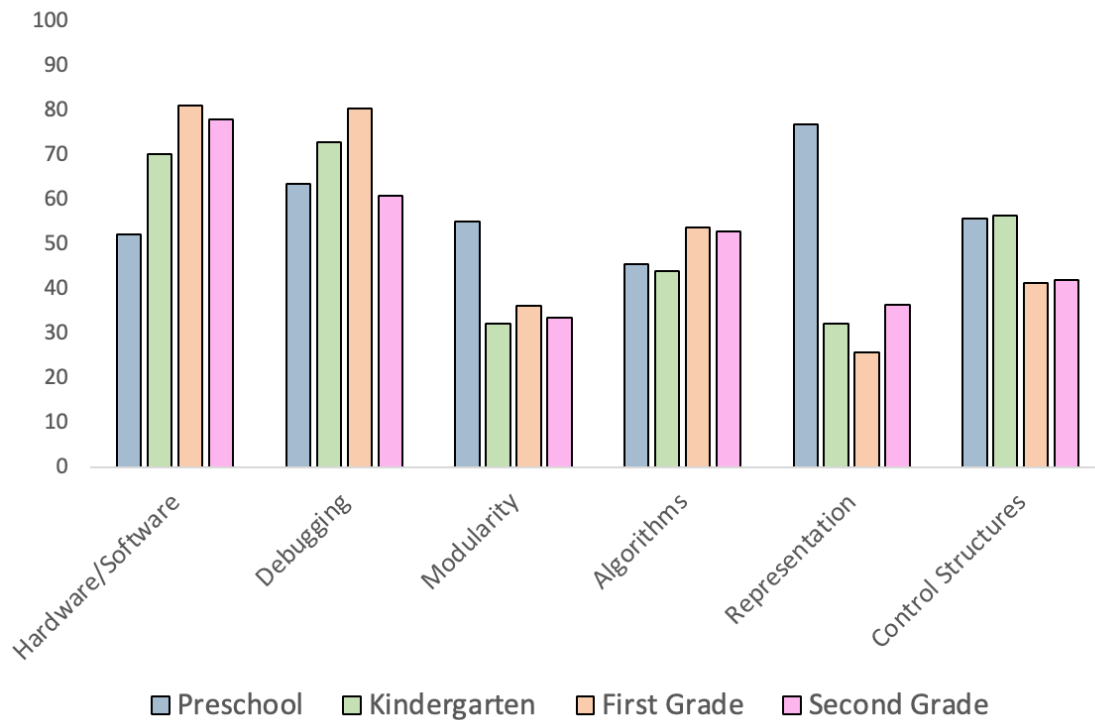
**Table 11***RQ 1 Part 2 Demographics by Grade*

	<b>PreK</b>	<b>K</b>	<b>1st</b>	<b>2nd</b>	<b>All data</b>
<b>Number of students</b>	173	395	935	618	2121
<b>Mean Age</b>	4.02	5.86	6.50	7.81	6.36
<b>SD</b>	0.62	0.42	0.56	0.35	1.15
Missing data	2	55	185	299	541
<b>Gender</b>					
Male	82	164	399	162	807
Female	84	171	400	157	812
Missing data	7	60	136	299	502
<b>Race</b>					
Black/African American	96	34	154	20	304
Hispanic/ Latino	29	42	113	56	240
Biracial/Multiracial	7	23	42	15	87
White	23	220	399	206	848
Asian	7	12	30	16	65
Other	3	4	8	6	21
Missing data	8	60	190	299	556
<b>Disability</b>	7	38	77	33	155
Missing data	93	70	244	299	706

**Results (RQ1 Part 2)**

The percent correct by domain for students receiving all four versions of *TechCheck* appropriate for their grade are shown in Figure 6.

To establish whether the scores were statistically different within and across domains, I conducted a crossed random-effects multilevel model using REML estimation. In this model, domain and subject (individual) were crossed and grade was a fixed effect predicting the domain score. I started with an empty model with percent of questions correct within each CT domain as the outcome variable and a random effect of CT domain. The Intra-Class Correlation (ICC) was .27, which indicates that about 27% of the variation in CT domain

**Figure 6***Baseline TechCheck Domain scores by grade*

percent correct was between domains (with the remaining percentage being differences between students across domains). I then added in a random effect of the intercept for the student variable. The deviance significantly decreased and the likelihood ratio test was significant, indicating the model with a random effect for both domain and student had a better fit ( $\Delta\chi^2(1) = 377.31, p < .0001$ ) (in other words, there were overall differences between students when considering all domains together). Lastly, I added the predictor of grade (type of assessment administered). This addition did not decrease deviance in the model, indicating that there is no difference across domains by grade. Upon examining the random effects, between-subjects variance attributable to the student and domain was .01 and .04 respectively,



indicating only a small contribution to the total variance (see Table 12). The ICC of .36 suggests approximately 36% of the variation is between domains.

**Table 12**

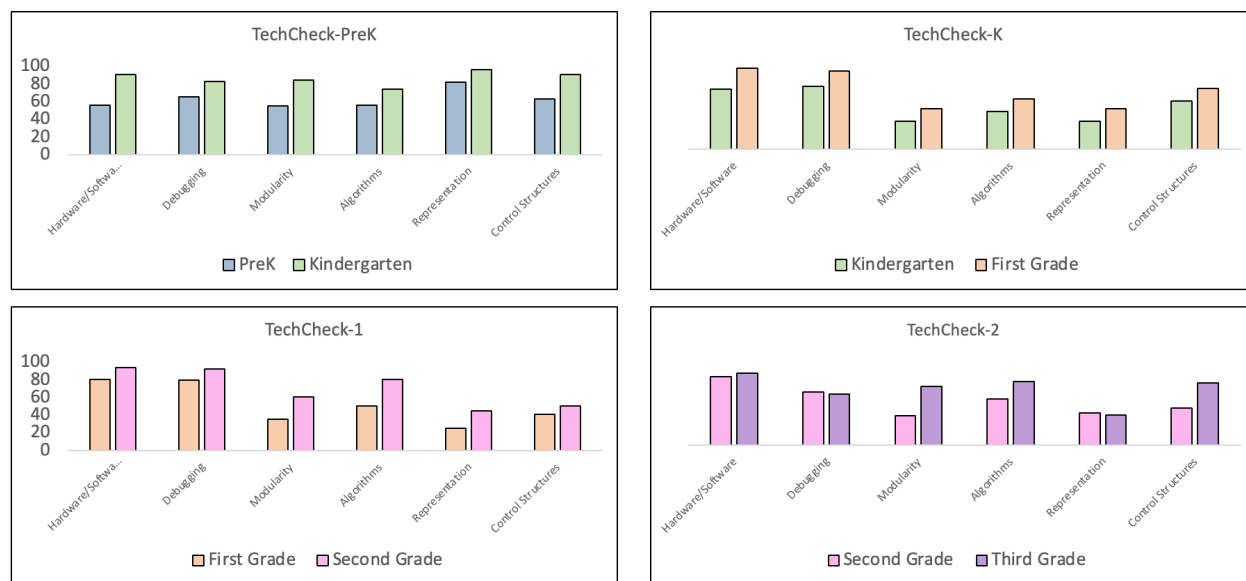
*Random Effects Table for Cross-random Effects Model Examining Baseline CT Domain Scores*

<b>Random Effect</b>	<b>Parameter</b>
$\tau_{00}$ (Student)	.01
$\tau_{00}$ (Domain)	.04
$\sigma^2$	.09
<b>ICC</b>	.36
<b>Deviance (-2LL)</b>	6440.6

Having established that the four versions of *TechCheck* had comparable within-domain performance across grades, I next analyzed performance by domain in the grade pairs. Figure 7 shows the percentage of questions in each domain answered correctly for each of the grade pairs. In most cases, a greater percentage of correct responses were made by children in the higher of the two grades in the pair. Overall, differences from one grade to the next appears to be fairly uniform across all CT domains with the exception of *TechCheck-2* which shows some variability. The data were noisier for the *TechCheck-2* pair, likely because there was a very small difference in age between the second and third graders and because a majority of third graders were English Language Learners (52.78%).

**Figure 7**

*Grade Pair Performance by Domain for the Four Grade-specific Versions of TechCheck*



**Research Question 2: Which demographic and environmental factors predict baseline CT performance in children ages 2-9?**

### Participants (RQ2)

Research question two included the same subset of  $n = 2121$  participants that were used in Research Question 1 part 2. Demographic information and descriptives are presented above in Table 11.

### Results (RQ2)

To further characterize the development of CT variables related to CT in young children, I conducted Linear Mixed Modeling (LMM) to explore demographic and environmental factors that could predict *TechCheck* performance in coding-naive students. Children with missing information about age, gender, or race/ethnicity were excluded from this analysis. I used the baseline *TechCheck* raw scores from all eight studies as the outcome variable and included in the model the variables of age, grade, gender, disability status, race/ethnicity as predictors with a

random intercept for classroom/teachers. A null model was first estimated and from that, an Intra-Class Correlation (ICC) of .12 was calculated. This indicates that 12% of the variation of student *TechCheck* scores was between classrooms/teachers.

To explore whether the variables age, grade, gender, disability status, and race/ethnicity predicted *TechCheck* scores, I added in each variable as a fixed effect sequentially. Using a likelihood ratio test, I found that a model with the fixed effects of gender and disability status did not improve the fit relative to the null model ( $\Delta\chi^2(2) = 3.39, p = .18$ ). Therefore, those two variables were dropped from the model. The model now included age, grade, and race/ethnicity as fixed effect variables and teacher with a random effect of the intercept. Next, I conducted a likelihood ratio test comparing a model that added a term for the interaction of grade and age to the model without that interaction term. The model with the interaction term had a significantly better fit ( $\Delta\chi^2(3) = 11.74, p < .001$ ). Relative to the first grade reference group, grade (preschool and kindergarten) were significant negative predictors of *TechCheck* outcome. The interaction of age and grade (preschool, kindergarten, and first grade) was a positive predictor in the model. This suggests that on average older students in each grade tended to score higher than their younger counterparts. Table 13 shows the results from the final model in more detail.

To provide a further measure of the robustness of the LMM model in RQ2, the same analyses were replicated using percentile ranks instead of *TechCheck* raw scores. The results using percentile ranks were nearly identical to those obtained with the previous raw score model in RQ2, suggesting that the model was not compromised by differences in the baseline score distributions for the respective versions of *TechCheck*. The equivalence of results supports the hypothesis that grade and age are genuine predictors of baseline *TechCheck* performance, not an

artifact of differences in the versions of the *TechCheck* assessment. Figure 8 shows a comparison of the magnitude of fixed effects (95% C.I.) when raw and normalized *TechCheck* scores were used as the outcome variable.

**Table 13**

*Final Model Results Predicting TechCheck Baseline Raw Scores*

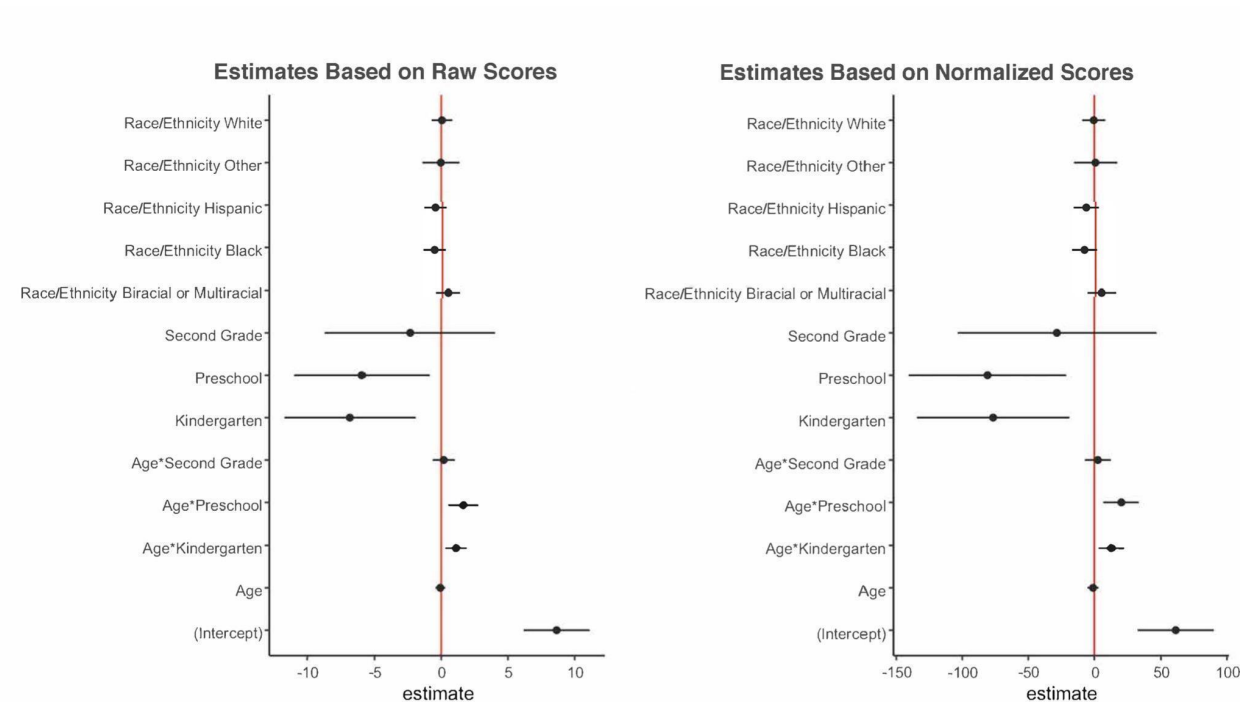
Parameter	Estimate	Standard Error	T-value	DF	P value	95% Confidence Interval	
						Lower Bound	Upper Bound
White	0.05	0.40	0.14	1350.83	$p = .05$	-0.73	0.83
Other	-0.03	0.71	-0.04	1346.68	$p = 0.97$	-1.42	-1.36
Hispanic	-.53	0.43	-1.24	1357.14	$p = 0.22$	-1.37	0.31
Black	-.59	0.43	-1.39	1358.73	$p = 0.17$	-1.43	0.25
Biracial or Multiracial	.44	0.48	0.92	1341.76	$p = .36$	-0.49	1.37
Second Grade	-2.33	3.24	-0.72	1356.98	$p = .47$	-8.76	4.11
Preschool	-5.92	2.58	-2.30	1347.57	$p = .02$	0.465	2.71
Kindergarten	-6.81	2.49	-2.73	1339.44	$p = .01$	-11.80	-1.81
Age*Second Grade	0.19	0.43	.44	1355.77	$p = .66$	-0.66	1.04
Age*Preschool	1.58	0.57	2.77	1342.21	$p = .01$	0.46	2.71
Age*Kindergarten	1.01	0.41	2.45	1355.57	$p = .01$	0.21	1.82
Age	-0.07	0.18	-0.38	1007.27	$p = .70$	-0.43	0.29
Intercept	8.65	1.26	6.86	1023.99	$p < .0001$	6.18	11.12

Random Effect (Teacher/Class)	Parameter
$\tau_{00}$	0.38
$\sigma^2$	5.41
Deviance (-2LL)	6217.0

*Note. First grade and Asian were used as reference group for the fixed effects of grade and race*

**Figure 8**

*Comparison of Fixed Estimates in LMM by Scoring System*



The LMM did not show a significant difference based on race/ethnicity. However, there were limited numbers of students within many of the race/ethnicity categories. To explore this variable further, I recoded the race/ethnicity variable into two categories: white and non-white. The mean total *TechCheck* score of the white category was 7.82 while that of the non-white category was 7.72. Unadjusted means for the two race/ethnicity categories were not significantly different  $t = 0.76$ ,  $df = 1534$ ,  $p = .45$ .

**RQ 3 How do coding educational interventions (specifically CAL-ScratchJr, CAL-KIBO, and codeSpark) affect the rate of acquisition of CT skills overall and in select subdomains?**

***RQ 3 Part 1: How do coding educational interventions affect CT in young children?***

**Participants (RQ3 part 1)**

The effects of coding interventions were examined using data from three completed studies involving first graders who were administered the assessment before and after learning to code. No other grades were included in this analysis owing to lack of availability of suitable datasets. A control group of students who participated in regular classroom activities without engaging in coding from the CAL-KIBO study was included in the analyses. Demographics of the respective study populations are shown in Table 14.

**Table 14**

*Demographic Information for Coding Interventions and a Control Group (RQ3 part 1)*

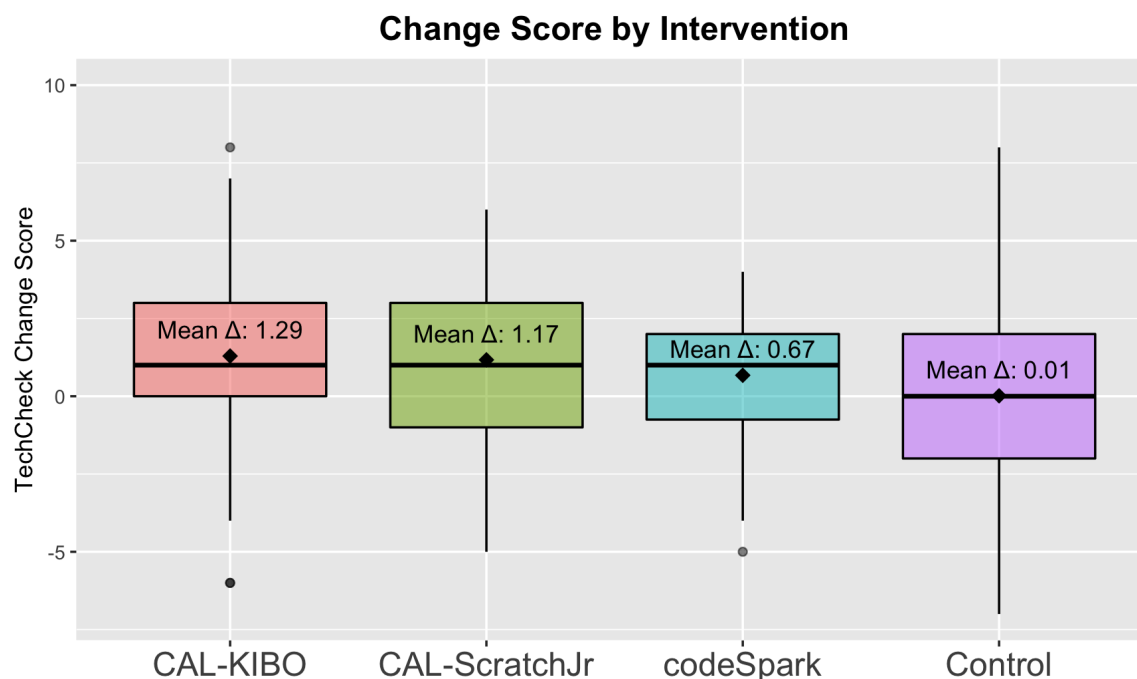
	CAL-KIBO	CAL-ScratchJr	codeSpark	Control	All data
Number of students	273	88	46	71	478
Mean Age (SD)	6.23 (0.51)	-	6.06 (0.23)	6.28 (0.45)	6.23(0.48)
Missing data	0	88	1	0	89
Gender					
Male	133	38	27	31	229
Female	138	38	19	40	235
Missing data	2	12	0	0	14
Race					
Black/African American	96	-	0	41	137
Hispanic/ Latino	28	-	31	12	71
Biracial/Multiracial	26	-	0	4	30
White	110	-	1	13	124
Asian	7	-	2	0	9
Other	4	-	1	1	6
Missing data	2	88	11	0	101
Disability	26	-	-	7	33
Missing data	0	88	47	0	135

### Results (RQ3 part 1)

Figure 9 shows the longitudinal change scores of three coding interventions on *TechCheck* total scores.

**Figure 9**

*TechCheck Change Scores for Three Coding Interventions and a Control Group*



Paired sample *t*-tests were carried out to examine whether children's scores changed significantly as a result of participation in a coding intervention. Cohen's *d* effect sizes were also calculated as a measure of the magnitude of differences in growth scores by each of the interventions. Table 15 shows *t*-test and effect size results by intervention. The effect size within the control group was considered negligible. All of the coding interventions had effect sizes within the small to medium ranges (Romano et al., 2006).

There were differences in the durations of respective interventions. The average length of time between pre and post-test was 60 days for CAL-KIBO, 89 days for Scratchjr, 64 days for codeSpark, and 54 days for the control group. Owing to the lack of availability of data on the precise number of hours of participation, no adjustment of intervention hours was carried out.

**Table 15**

*Magnitude of Effect on CT Skills by Coding Intervention and a Control Group*

<b>First Grade Intervention</b>	<b>T-value</b>	<b>Significance</b>	<b>Cohen's <i>d</i> (Effect Size)</b>
CAL-KIBO	$t(272)=9.28$	$p < .0001$	0.54
CAL-ScratchJr	$t(87)=4.52$	$p < .0001$	0.47
codeSpark	$t(46)=2.05$	$p = .04$	0.25
Control	$t(70)=0.04$	$p = .97$	0.01

LMM was conducted with data from first grade students to examine how different interventions affect student learning of CT skills. The change in *TechCheck* total score (baseline subtracted from end-point) was the outcome variable. I started by fitting an empty model with *TechCheck* change score as the outcome variable and teacher/classroom as a random effect of the intercept. The ICC value for the empty model was .05 indicating approximately 5% of variation in the change scores is due to the child's class membership. Next, I added in type of intervention (CAL-KIBO, codeSpark, CAL-ScratchJr, or a control group) as a fixed effect variable, which resulted in an improved model fit ( $\Delta\chi^2(3) = 12.09, p = .007$ ). I then added in the child's baseline *Techcheck* score to account for baseline score differences. Once again, model fit significantly improved ( $\Delta\chi^2(3) = 137.04, p < .0001$ ). Gender and duration of the intervention did not contribute significantly to the model. These two variables were therefore removed. The model with the best Akaike Information Criterion (AIC) value was one with a random effect of the intercept for teacher/classroom. The final model included the outcome variable of *TechCheck* change score with the predictors of type of intervention, baseline score, and a random effect of the intercept of classroom/teacher.



Table 16 shows the estimates obtained with the final model. Among the interventions, only CAL-KIBO was found to be a significant predictor of *TechCheck* change with the control group as a reference group in this model.

**Table 16**

*Final Model Results Predicting TechCheck Change Score by Intervention*

Parameter	Estimate	Standard Error	t-value	DF	p value	95% Confidence Interval	
						Lower Bound	Upper Bound
CAL-ScratchJr	0.86	.45	1.91	22.04	.05	-0.03	1.75
codeSpark	0.80	.52	1.54	20.91	.12	-0.22	1.82
CAL-KIBO	1.51	.37	4.11	21.06	<.001	0.78	2.23
Baseline <i>TechCheck</i> score	-0.53	.04	-12.65	459.00	<.001	-0.61	-0.45
Intercept	4.29	.47	9.09	70.04	<.001	3.36	5.22

### Random Effects

$\sigma^2$  4.27

$\tau_{00}$  Teacher 0.20

ICC 0.04

$N_{\text{Teacher}}$  34

---

Observations 464

Marginal  $R^2$  / Conditional  $R^2$  0.280 / 0.312

### **RQ3 Part 2: Coding educational interventions and CT domain-specific TechCheck Score**

#### **Participants (RQ3 part 2)**

RQ3 part 2 was addressed using the same data as RQ3 part 1. See Table 14 for details.

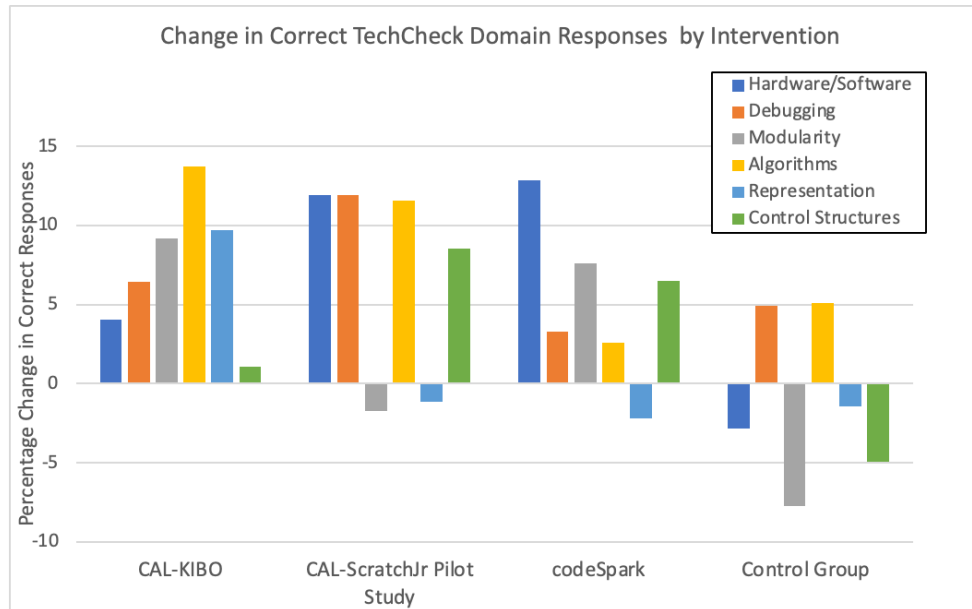
## Results (RQ3 part 2)

To examine performance by domain in first grade, the change in the percentage of correct responses within each domain was calculated for each of the three coding interventions (see Figure 10). To assess whether the domain change scores for first grade were statistically different from each other, I estimated another crossed random-effects multilevel model with the same sources of variation as Research Question 1 Part 2.

To establish whether there is a difference within and across domains by intervention, I conducted a cross random effects model using ML estimation. I started with an empty model with each participant's change in percent of questions correct per CT domain as the outcome variable and a random effect of the intercept for the type of domain. Next, I added a random effect of the intercept for the student variable which significantly reduced the deviance ( $\Delta\chi^2(1) = 7.13, p = .008$ ), which indicates there were between-student differences in overall scores across all domains. In the model which did not include coding intervention as a variable, the ICC for students was .03 and the ICC for domain was 0. Those values indicate that little to none of the variation in change in CT domain score was between students or domains. I then added in a fixed effect for the intervention the child received (CAL-KIBO, CAL-ScratchJr, codeSpark, or a control group participating in everyday classroom activities). The deviance significantly decreased and the likelihood ratio test was significant, indicating the model with the intervention variable had a better fit ( $\Delta\chi^2(3) = 34.76, p < .0001$ ). Results of the model are shown in Table 17.

**Figure 10**

*Change in CT Domain Scores Across Coding Interventions and a Control Group*

**Table 17**

*Estimates from LMM of Domain Change Score by Coding Intervention*

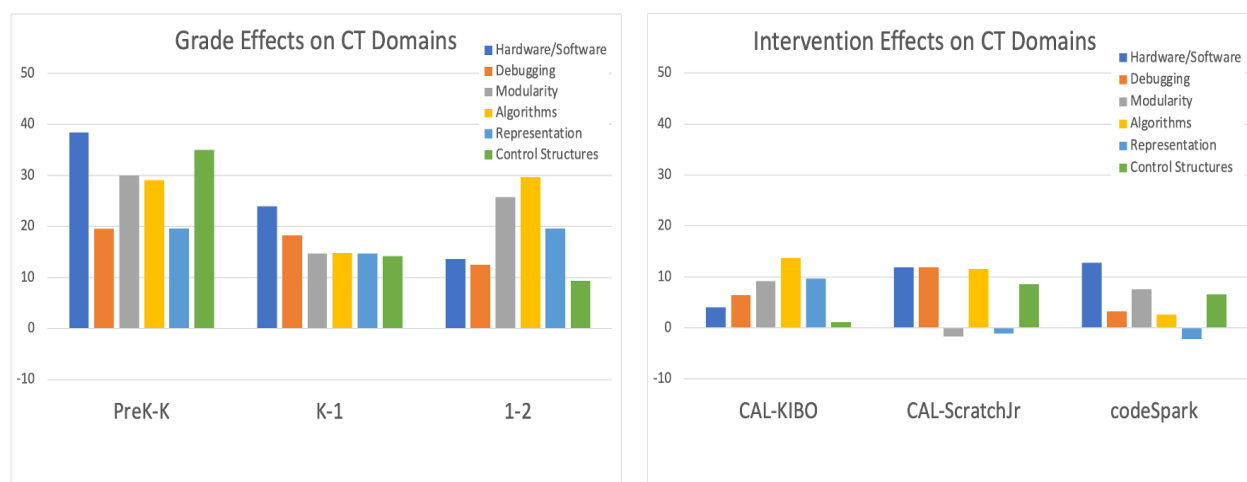
<i>Predictors</i>	<i>Estimates</i>	<i>CI</i>	<i>p</i>
<i>(Intercept)</i>	<i>-0.01</i>	<i>-0.05 – 0.03</i>	<i>0.596</i>
<i>ScratchJr</i>	<i>0.08</i>	<i>0.03 – 0.13</i>	<b><i>0.002</i></b>
<i>codeSpark</i>	<i>0.06</i>	<i>0.00 – 0.12</i>	<b><i>0.040</i></b>
<i>CAL-KIBO</i>	<i>0.09</i>	<i>0.04 – 0.13</i>	<b><i>&lt;0.001</i></b>
<b><i>Random Effects</i></b>			
$\sigma^2$	<i>0.13</i>		
$\tau_{00}$ Student	<i>0.00</i>		

$\tau_{00}$ Domain	0.00
ICC	0.03
$N_{Domain}$	6
$N_{Student}$	478
<hr/>	
Observations	2867
Marginal $R^2$ / Conditional $R^2$	0.006 / 0.038

Finally, I prepared a plot showing the differences between grades on *TechCheck* domain scores based on data from the RQ1 grade pair analysis compared to the effects of coding interventions as analyzed in RQ3 (see Figure 11). It is evident that the magnitude of improvement is greater for grade than coding interventions across all domains. This finding is not unexpected since the coding interventions are relatively brief compared to the interval of one grade level. It is also notable that the effects of grade are relatively consistent across CT domains whereas the interventions appear to have more selective effects on particular domains.

**Figure 11**

*Comparison of the CT domain score differences by grade and coding intervention*



## Discussion

Although there has been increasing interest in CT over the past several decades, there is still only a limited understanding of how young children develop these thinking skills. In this dissertation, I attempted to address three research questions relating to the acquisition of CT skills in early childhood. I first examined *TechCheck* performance in coding naive children across several grades to gain insights into CT skill acquisition in the absence of coding instruction. Next, I explored whether demographic and environmental factors predict CT performance in children between 3 and 9 years of age. I then explored whether CT skills acquired in the course of normal development differed from those that were obtained following various coding/CS educational interventions. The studies in which educational interventions were implemented also provided an opportunity to compare the effects of different coding interventions on CT skills, which has not been extensively studied in children in this age group. The implications of the outcomes I obtained for the three research questions are discussed below.

### **RQ1: To what extent do CT skills differ in coding-naive children in preschool through third grade?**

One of the principal aims of this study was to explore how young children's CT skills change in the absence of coding instruction. I approached this by administering grade-specific versions of *TechCheck* to groups of coding-naive children and comparing their performance to that of coding-naive students in one grade higher who were given the same version of *TechCheck*. This design was inspired by an earlier study examining the longitudinal impact of the CAL-KIBO curriculum on the CT skills of first and second graders. At the time that study was carried out, the original *TechCheck* (*TechCheck-1*) was the only version available. It was therefore administered to both first and second graders. To provide a yardstick for understanding

the magnitude of the change in *TechCheck* scores attributable to the coding intervention in that study, I calculated the mean difference in baseline *TechCheck* scores and divided it by the mean difference in age between first and second graders. That calculation was used to estimate the change in *TechCheck* scores associated with the six to seven week-long CAL-KIBO curriculum. The calculation suggested it was approximately equivalent to the change occurring over a span of six months of normal development (Relkin et al., 2021). A similar approach was taken by Arfé et al., (2020) in measuring the effects of a coding intervention on neuropsychological test results.

The present study extended this design to five grades (pre-K, K, 1, 2, 3) and used four grade-specific versions of *TechCheck* (*TechCheck* pre-K, K 1 and 2). All grades had a normal distribution of scores, which was important to confirm as a prerequisite for the parametric statistical modeling I carried out on the assessment data. I found the mean *TechCheck* scores to be significantly greater in the higher grade of each pair. The magnitude of the differences in the means between grades, even after age adjustment, was greater in this study than that observed between first and second graders in the earlier study by Relkin et al., (2020). This discrepancy may in part be related to differences in the samples enrolled in the respective studies. Regression analysis using age as the independent variable did provide estimates of *TechCheck* score change per year that were closer to those obtained in the earlier study. It is possible that the use of means alone as measures of central tendencies in the grade pair analyses was inadequate. Regression analysis adjusted for age may provide a more accurate estimate of change in CT skills across grades than taking differences in age-adjusted means.

Although *TechCheck* has not been validated as a measure of performance within individual CT domains, the association of each question with a particular CT domain was previously confirmed by expert consensus (Relkin et al., 2020). It was therefore reasonable to

carry out exploratory analyses using groups of related questions as probes of each of six CT domains. Analysis of the baseline *TechCheck* data revealed that the percentage of correct responses by domain in pre-K through second grade were uniformly higher in all six domains in the upper of the two grades in each pair (see Figure 7). The second versus third grade pair did show some inter-domain variability but these results were likely influenced by the make-up of the third grade cohort. That group was relatively small and contained a sizable percentage of English language learners.

The uniform increase in CT skills across the six domains in Pre-K through second grade is an interesting and somewhat unexpected finding that contrasts with the results I obtained from longitudinal studies of the effects of the three coding educational interventions (see RQ3 discussion below). Because students in this age range are transitioning from a preliterate stage to early literacy, one might expect to see more selective improvements in certain domains such as representation, which is exercised extensively in learning to read and write. Bers (2020) has outlined the possible relationship between the acquisition of literacy and each of the powerful ideas of CS. However, I found no evidence of a selective effect on representation or the five other CT domains when comparing pre-literate (eg: pre-K) to literate (eg: second grade) coding-naive students.

One possible explanation for the broad improvement in *TechCheck* scores across domains with advancing grade is the effects of brain maturation. A young child's literacy, numeracy, and abstract reasoning are known to undergo gradual development as they grow (Piaget, 1971). The period of growth between the ages of three through nine is one of rapid cognitive development paralleled by changes in the functional connectivity within the brain (Gerdes et al., 2013; Vogel & Smedt, 2021). There is preliminary evidence that *TechCheck* scores correlate with certain

executive function measures (M. Willoughby, personal communication, February 14, 2022). Past studies have found that CT skills correlate with various neuropsychological measures such as fluid intelligence, working memory, planning, sequencing, mental rotation, vocabulary, and early math precursors such as numerical transcoding and symbolic magnitude comparison (Arfè et al., 2020; Gerosa et al., 2021). Neuropsychologists use age-adjusted norms to interpret these measures, indicating that they are subject to change as a function of age in the course of normal development.

Abilities such as reading and writing (and likely CT) are mediated by multiple brain regions that originally evolved to serve other purposes (Peters & Smedt, 2018). As a consequence, exercise of these abilities requires the coordinated activity of multiple brain regions (Fias et al., 2013). As the brain progressively myelinates in the decades after birth, communication between interconnected brain areas becomes more efficient. This may particularly enhance cerebral functions that are spatially distributed in the brain and are therefore dependent on signal transmission between different regions. The uniform increase across all CT domains with advancing grade observed in the present study could be consistent with a generalized effect of brain maturation on cognitive skills in early childhood. Future studies should examine the extent to which the improvement in CT with advancing grade correlates with other cognitive abilities as well as measurable aspects of brain development, such as white matter myelination seen on MRIs (Chrysikou, et al., 2022).

Another factor that could contribute to improvement in CT in coding-naive students as they advance in grade is cumulative learning experiences. Various scholars have emphasized that children learn as a product of engagement with their environment and introduction to new materials (Odegard, 2012; Penfold, 2019). It is possible that coding-naive students' baseline



performance on *TechCheck* reflects their cumulative exposure to everyday experiences, including engagement in solving day-to-day problems that are analogous to *TechCheck*'s unplugged challenges. This possibility cannot be readily ruled in or ruled out from the available data since there are no included measures of everyday experience. To help discern whether children acquire CT skills from everyday challenges, future studies might examine the correlation between *TechCheck* scores and specific life experiences such as participation in summer camp activities that expose children to different challenges than they confront during the regular school year. It may also be useful to examine whether children growing up in environments with less resources (eg: lower SES) have lower baseline scores on *TechCheck* than those growing up in more privileged circumstances.

## **RQ2 Which demographic and environmental factors predict baseline CT performance in children ages 2-9?**

The second research question examined possible intrinsic and extrinsic predictors of *TechCheck* scores in coding-naive young children. Using LMM analyses, only grade was found to be a predictive variable. I did not find significant predictive effects for age, gender, race/ethnicity or disability. An interaction was observed between age and grade, suggesting that within a given grade, older children tend to score higher than younger ones. Age was not a significant predictor on its own in models that included grade as a variable.

Females are historically underrepresented in the field of computer science (Kanaki & Kalogiannakis, 2022). Some past studies have shown that children can develop gender stereotypes about technology that can influence their acquisition of coding and CT skills (Jenson & Droumeva, 2016; Sullivan, 2016; Sullivan & Bers 2016). Others have found that there is no gender difference in coding or CT skills in young children (Kanaki & Kalogiannakis, 2022;

Papavlasopoulou et al., 2020). Despite a fairly robust representation of male and female students in the current study, no indications of a difference in CT skills by gender were found. This finding does not diminish the importance of initiatives that promote gender equality in STEM and coding education, such as the National Girls Collaborative, National Center for Women & Information Technology and Girls Who Code. If successful, such initiatives can help to reduce the gender disparities that persist in computer science and related fields.

Likewise, the lack of a predictive effect of race/ethnicity in the present study should not be taken as conclusive proof of the absence of such effects. Previous researchers have reported that some children from racial minority groups have limited access to technologies (Margolis, et al., 2017; Wang & Hejazi Moghadam, 2017). Other studies have found that race can be a proxy measure for other factors such as SES (Williams et al., 2016). The lack of an observed effect of race/ethnicity in the current study neither supports nor rules out a relationship between race and CT since the cohort is not a representative sample of minority groups nor was the study necessarily powered to examine the effects of race/ethnicity on CT. There may still be differences in CT skills related to these demographic variables as well as related to factors such as differential access to technology. Future studies should incorporate race, SES and measures of technology access to further examine how these variables are related to CT outcomes.

The present study did not find disability to be a significant predictor of *TechCheck* performance in coding-naive students. Data on the disability status of participants were incomplete, which may have contributed to the lack of an observed effect. Specified disability numbers were often small and several sites chose not to share information about specific types of disability out of concern about identifying specific children. Other studies have shed light on the importance of inclusive initiatives and curricula for those with disabilities such as

AccessCSforAll (Ladner, & Stefik, 2017). Recently, Levinson and Bers (2022) found that coding naive students diagnosed with disabilities had lower baseline *TechCheck* scores than students without disabilities, but improved on *TechCheck* to a comparable extent as non-disabled students following a coding intervention.

**RQ3 How do coding educational interventions (specifically CAL-ScratchJr, CAL-KIBO, and codeSpark) affect the rate of acquisition of CT skills overall and in select subdomains?**

The approach to the third research question involved analyzing data on changes in *TechCheck* scores from students engaging in one of three coding curricula. Results support the hypothesis that CS educational initiatives that promote learning to code can accelerate the acquisition of CT. In particular, in children receiving the CAL-KIBO curriculum, *TechCheck* total scores increased significantly more than in students carrying out regular classroom activities over a comparable time period. Although the ScratchJr and codeSpark studies did not include control groups, *TechCheck* scores increased significantly compared to baseline after each of these coding interventions. Although the effects of all three coding interventions were smaller in magnitude than the changes observed between grade pairs (see Figure 11), this finding is not unexpected given that the coding interventions were relatively brief (6-12 weeks) compared to the age differences between grades in the grade-pair analyses.

This study is also notable for providing some of the first (preliminary) evidence that different coding curricula involving tangible (i.e. KIBO) or screen-based (i.e. ScratchJr and codeSpark) coding platforms may impact CT domain scores in different ways. Specifically, different domain-specific patterns of response were observed for the three coding interventions (see Figure 10). Keeping in mind caveats about the exploratory nature of the CT domain analysis, one can speculate about possible factors contributing to the differences in domain

change scores across the three coding interventions. Although children engage with all of the powerful ideas probed by *TechCheck* when coding with all of these platforms, there are differences in the manner in which they are engaged. One difference is the use of tangible versus screen-based programming platforms. Another difference is in the grade-appropriateness of the content and differences in emphasis on specific CT domains with different coding curricula. A third consideration relates to the unplugged nature of the questions on *TechCheck* and the extent to which the three curricula include similar unplugged activities.

A possible difference in the impact of tangible versus screen-based platforms may be exemplified by the results obtained in this study in the domain of hardware/software. The percentage of correct responses on hardware/software challenges were numerically lower in students who learned to code with the tangible KIBO platform than with the screen-based ScratchJr and codeSpark applications. KIBO's software does not have to be loaded by the child and its coding blocks are actual physical entities. This could lead to some confusion about the difference between hardware and software. On screen-based platforms such as ScratchJr and codeSpark, coding elements appear only after the software application is loaded on a hardware device. The only physical manipulatives for those platforms are the computer/tablet running the software and the input device, which may be the screen or a mouse. It is possible that as a result of these differences, some children learning to code on a tangible platform such as KIBO less readily grasp the distinction between hardware and software than those learning coding on screen-based platforms.

Students receiving the CAL-KIBO curriculum had fewer correct responses in the domain of control structures compared to students receiving CAL-ScratchJr or codeSpark. All three curricula teach children about loops, conditionals, and nested statements, which are important

elements of control structures. However, KIBO's conditionals are limited to eight "If, End-If" or "Repeat-Until, End-Repeat" commands that have a relatively high level of difficulty for young children. These commands are consequently introduced later in the CAL-KIBO curriculum at about the 9th lesson (out of 12 total lessons). The types of conditionals and events offered by the two other curricula have a greater range of complexity and are implemented sooner than the CAL-KIBO curriculum. For example, ScratchJr's "Start on Tap" command is typically considered to have an easy to medium level of difficulty. In contrast, ScratchJr's "Start on Orange Message" commands can be difficult for young children to comprehend. Likewise, children may find codeSpark's "Start on Bump" blocks to be easier to grasp than the more advanced "If Hungry" commands. It is conceivable that some of the conditionals introduced by the screen-based curricula were easier to master than those introduced by CAL-KIBO, resulting in the observed difference in control structure domain scores across curricula.

Different coding curricula emphasize certain concepts to different degrees. Students engaging with codeSpark had the highest percentage of correct responses in the hardware/software domain. Perhaps this finding is due to the codeSpark curriculum explicitly teaching about the meaning of "computers" and "programming" and how different smart devices are programmed. In contrast, the majority of CAL-KIBO and CAL-ScratchJr lessons that were designed to teach about hardware/software focused on enhancing the child's competency in using the coding platform (e.g., understanding KIBO's sensors; learning how to open up a new project on ScratchJr). Less emphasis was placed on defining computers, programs, or understanding how smart objects work. However, in the CAL curricula, other concepts not assessed by *TechCheck* were introduced such as comparing and contrasting human languages vs. programming languages.

The coding projects that children were asked to engage in during CAL-KIBO and CAL-ScratchJr interventions were open-ended in nature, allowing children to use any coding blocks they wished. In addition, the KIBO and ScratchJr platforms were designed to have a “low floor, high ceiling, wide walls” ( Grover & Pea, 2013; Resnick & Robinson, 2017). This meant that the platforms were simple enough for a beginner to engage with but also could become challenging for more advanced users. Although there were often multiple possible solutions to advance through the codeSpark puzzles, they typically required children to use a set amount of coding blocks in a particular order. Children in first grade are still developing working memory skills and it is possible that some of the codeSpark puzzles contained algorithms that were too lengthy for this age group. In fact, there is some preliminary evidence that third graders presented with the same puzzles improved on *TechCheck* at a much higher rate (Iseli et al., 2021a).

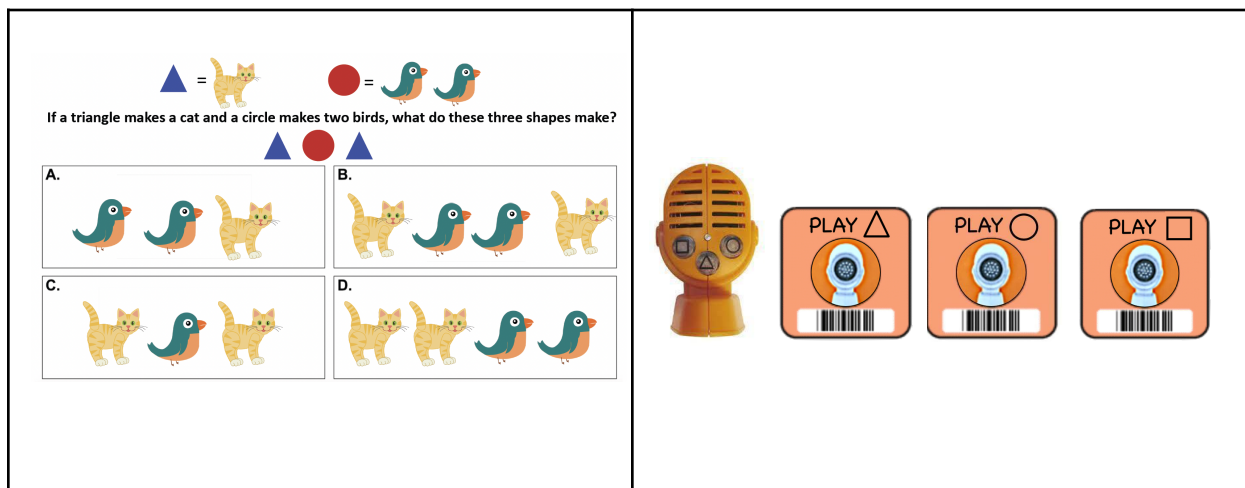
One lesson in the CAL-KIBO curriculum also coincidentally resembled elements of the representation questions on *TechCheck*. The lesson involved the child learning about KIBO’s sound recorder module that allows children to record up to three different sounds by pressing either a triangle, circle, or square. Students played a “shape-shifting” game in which volunteers chose a sound or action to go with each shape. For example, one student might decide to jump on one foot for the circle, another student might decide to yell “Hooray!” for the Triangle, and a third student might decide to hold up a book for the Square. To play the game, when the teacher held up one of the three symbols, the class would only perform the action associated with that symbol’s shape. Figure 12 shows a representation question from *TechCheck* next to the sound recorder module and sound symbol blocks. The child needs to exercise a very similar kind of thinking and problem solving skills to succeed in the “shape-shifting” game as they do to get the

representation questions on *TechCheck* correct. The “shape-shifting” game could help explain why children in the CAL-KIBO group scored higher in the representation domain than children learning to code with the ScratchJr or codeSpark apps.

There has been a paucity of past studies comparing the impact of different coding interventions on the CT skills of young children or comparing different curricula’s effects on specific CT domains. Although the present study involves posthoc analysis of previously collected intervention data and uses an assessment instrument that was not optimally designed for CT domain analyses, these exploratory findings provide a rationale for future studies comparing CS curricula using the yardstick of unplugged CT assessment.

### Figure 12

Example of a Representation Question on *TechCheck* (left) compared to coding block used with KIBO’s sound recorder module (right).



### Limitations and Future Work

The primary aim of this study is to better understand the development of CT in young children. Although longitudinal intervention studies were included in the analyses, many of the conclusions were drawn based upon cross-sectional data from coding-naive students in different

grades. The analysis of CT performance in groups of students at a given time point is not equivalent to serial observations performed on the same children over time. The cross-sectional analyses provide information about group tendencies rather than individual development trajectories. The design and implementation of prospective longitudinal studies of the acquisition of CT will take many years and considerable resources to complete. It may not be feasible or ethical to have a control group of children who remain coding-naive for several years, bringing into question whether true randomized control trials can be performed. It is therefore imperative to examine all available data to generate plausible hypotheses about how children acquire CT skills. Although some of the data included in this work were collected specifically for this dissertation, I also drew upon data from several completed large-scale studies that were carried out for other purposes. Although this is an acknowledged limitation, I believe the exploratory studies I have carried out in this dissertation represent important groundwork for future investigations.

I have already pointed out that past studies using coding exercises to measure CT may conflate programming ability with CT skills. While the use of an unplugged assessment can help circumvent this issue, the use of unplugged challenges may introduce another potential confounder. Unplugged challenges, by definition, draw upon activities and experiences derived from children's everyday lives. It is therefore difficult to say whether the outcomes observed in the present study truly reflect the acquisition of CT skills or are measuring children's exposure to everyday experiences that resemble the types of unplugged challenges embodied in *TechCheck*. Stated another way, if coding challenges can conflate the measurement of CT skills and coding abilities, unplugged challenges may conflate the measurement of CT skills and exposure to everyday life experiences.



The cross-sectional method employed in this dissertation to answer RQ1 and RQ2 are subject to potential biases. For example, there may be other unmeasured demographic or environmental differences between the students in the respective grades (e.g., SES, certain types of disabilities) that influence their performance on *TechCheck* or alter the differences based on age in CT development. As such, these analyses must be considered exploratory and hypothesis-generating rather than conclusive measures of intra-individual change.

*TechCheck* was originally designed to be administered to groups of students working on computers or tablets in a classroom setting. This dissertation combines data from several studies in which the four grade-specific versions of *TechCheck* were administered in five different ways. Administration formats for this dissertation included in-person via tablet group administration, in-person via tablet one-on-one administration, in-person via paper and pencil format, virtual via one-on-one Zoom sessions, and virtual via Zoom group administration. These different modes of administration were necessary to accommodate school closures and remote learning during the COVID-19 pandemic as well as computer/staffing resource limitations at certain schools. A training and certification program for all *TechCheck* proctors was implemented to foster consistency regardless of the mode of administration. Because there was no direct comparison across modes of administration, it is possible that this introduced some additional variation in results. For example, the one-on-one virtual version of the assessment requires children to verbally indicate their answers. Thus that version may require more verbal skills and less manual dexterity than the version of the assessment in which children tap or circle their answer. Another possibility is that children assessed in groups may have been more distracted and/or less easily re-directed by a proctor than children in a one-on-one assessment session. Future studies should compare each method of administration to evaluate their equivalence.

This dissertation contains data collected prospectively as well as pooled re-analyses of previously collected data. The use of data from multiple studies increased power by expanding the number of participants available for analysis. It also allowed for comparisons across a broader range of grades than the individual studies would allow. However, the design, research questions and student cohorts from each of the contributing studies were somewhat different. There was also variation on the demographic and background data collected across studies.

It is possible that the COVID-19 pandemic impacted the study's outcomes. As a result of the COVID-19 pandemic, studies have shown that children are significantly behind in their academic performance as compared to pre-pandemic students (Dorn et al., 2020). Some of the data in this study were collected prior to the pandemic, while other data were collected during its course. Additional analyses using time of administration (pre or post pandemic) as a predictive variable may provide further information about the effects of the pandemic on the acquisition of CT skills.

Although this study includes a large sample of students from different U.S. states and demographic backgrounds, the results are not representative of all students in the target age range. To obtain a more representative sample and allow for truly standardized norms to be produced, this research should be replicated and extended to other cohorts. The *TechCheck* assessment has been translated into at least six different languages and is currently being administered in diverse research and educational settings in many countries around the world (e.g., Bosgoed & Fanchamps, 2022; Hançer, et al., 2021; Yang et al., 2020) Future studies should further explore its use in children from various cultures and in neuro-diverse children.

The analyses described in this dissertation have generated some interesting hypotheses about the processes that may be involved in the acquisition of CT in early childhood. However,

to draw robust conclusions and confirm results, the findings should be replicated with methodological adjustments. Sample size limitations as well as missing data likely affected the outcomes of analyses for all three of the research questions.

In examining the development of CT in young children for RQ1, cross-sectional data comparing different grades is not a satisfying substitute for longitudinal measurements in individual students. Ideally, the same children would have been assessed longitudinally at multiple time points as they progressed through successive grade levels, with careful attention paid to their CS and non-CS educational experiences over time. With a large-scale longitudinal study of CT development, it might be possible to identify “critical periods” of CT development in which the most change and/or domain-specific changes occur. This did not prove possible with the current study design.

Domain analysis was carried out in the present study for exploratory purposes. It is quite possible this was not a reliable way of measuring performance in these domains. *TechCheck* was not originally designed to measure performance in specific domains of CT. There are as few as two questions and as many as five questions per domain in this assessment. The number of questions per domain differed by version of *TechCheck* which may have added noise to the data. Prior research has shown that there is an increased level of reliability and construct validity when multiple items (above three) are used to measure a given construct (Diamantopoulos et al., 2012; Hair et al., 2010). Future studies might give students a greater number of questions to better probe each domain. Factor analysis or other methods could then be used to identify the subset of questions that provide a rapid and valid measure of performance within each CT domain.

RQ2 examined which demographic and environmental factors predicted baseline CT skills in children. A more ideal study might have included students from a sample with numbers

of each racial/ethnic category that are more representative of the US population as a whole. In a prospective study with balanced recruitment it may become possible to detect small effects and interactions that were inapparent in the current study.

RQ3 examined how coding educational interventions affect the rate of acquisition of CT skills. However, the participants in each study were somewhat different and each intervention spanned a different length of time (e.g., 12 hours for CAL-ScratchJr vs. 6 hours for codeSpark). Ideally, each intervention would have its own control group and samples would be matched in terms of demographics, baseline scores, and testing intervals.

Within the last few years, assessment of CT in early elementary school children has made considerable strides. Unplugged CT assessment offers some distinct advantages and is gaining increasing acceptance in the field. However, for the field to move forward, assessments such as *TechCheck* must be compared to other measures. Future work should include CT assessments of children using multiple measures such as *TechCheck*, the BCTt, cCTt, CTA, and/or CAPCT. Comparative studies represent the best way to determine the strengths and weaknesses of each measure. This comparison could help guide researchers and educators in choosing the best possible measures for their particular classrooms or studies.

No one measure can capture all there is to be learned about CT in young children. I advocate the use of a combination of assessment methods to get the most holistic interpretation of a child's CT skills. Other types of CT measures such as interviews, observations and game-based tasks can be helpful in determining a child's skill level. However, children should not be subjected to multiple forms of lengthy assessments that will take time away from their learning. In fact, to avoid just this, *TechCheck* was designed to be engaging and administered in 15 minutes or less. A useful model is one put forth by the CRESST laboratory at UCLA, which

combined the *TechCheck* assessment with naturalistic CT assessment via collection of real-time telemetry data from coding game-play (Iseli, et al., 2021a; Iseli et al., 2021b). The CRESST group formulated CT indicators that could be automatically scored as children are learning to code with the codeSpark application without any deviation from normal gameplay. It is not yet the case that educators and researchers can automatically collect meaningful naturalistic CT assessment data with strong validity and reliability evidence from young children's participation in activities such as coding. However, I believe in the near future many coding platforms for young children will move in this direction and incorporate naturalistic formative assessment. This type of assessment would allow us to track children's CT development in a more contextualized fashion and establish benchmarks of performance that can alert educators and researchers when a child is in need of extra support or would benefit from special enrichment.

## **Conclusions**

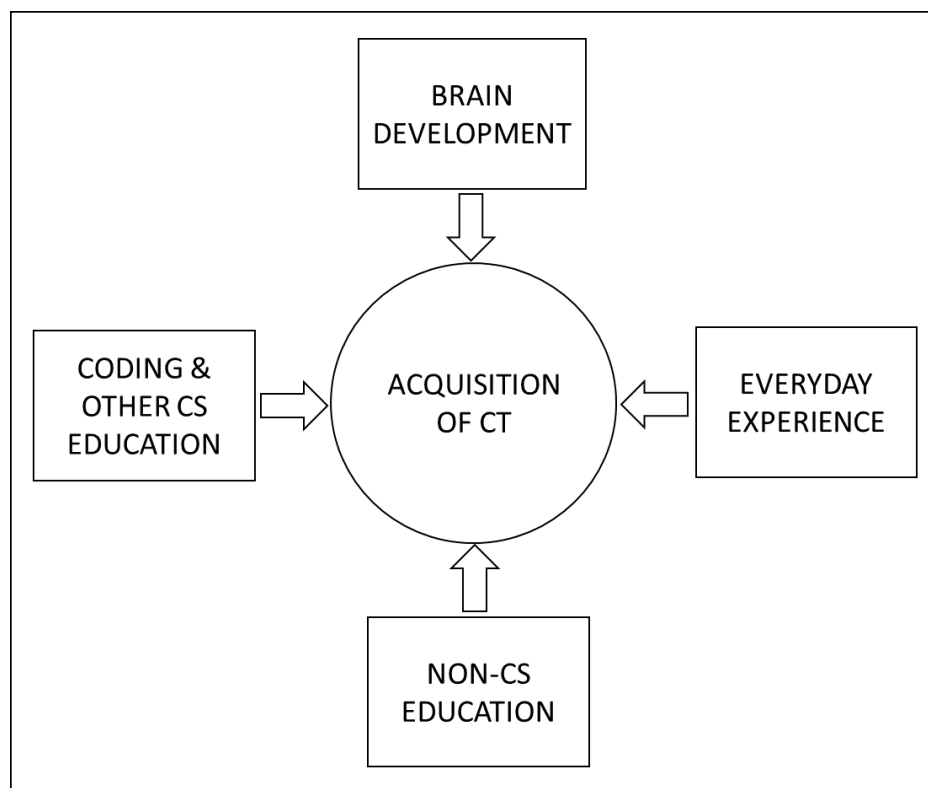
The advent of developmentally appropriate unplugged CT assessments for young children such as *TechCheck* has opened a new window on the study of CT skills in young children. Much of the current understanding about how children develop CT skills has been inextricably linked to coding and other elements of CS education. The extent to which everyday experiences and brain maturation influence the acquisition of CT skills has received much less attention. The availability of unplugged CT assessments makes it possible to examine how phenomena associated with brain development, everyday interactions with technology, non-technological experiences as well as CS and non-CS educational interventions influence the acquisition of CT skills as children grow.

The studies described in this dissertation represent initial steps in exploring this topic. The outcomes demonstrate that preschool and early elementary school children are capable of

engaging in aspects of CT that were once considered too complex and abstract for young learners. The study also establishes that unplugged challenges that exercise CT skills can be carried out by many children who have not learned to code. Furthermore, these CT skills can improve as children grow even in the absence of formal CS instruction. There is also evidence that early childhood CS educational interventions that teach coding can accelerate the process of learning CT in distinctive ways. Children may acquire basic intuitions about CT from learning to code, from their day-to-day experiences and from non-CS education. CT skills may also improve as a consequence of normal brain development (see Figure 13).

**Figure 13**

Processes that may influence the progressive acquisition of CT skills in young children



One of the tacit goals of this dissertation is to assist in identifying the best practices to enhance the acquisition of CT in young children. This work has helped to establish the utility of

unplugged CT assessment as a means for educators and researchers to measure and follow the progress of young children as they acquire CT skills (or fail to do so). It is my hope that CS education for young children can be improved by applying the information gained from this research. Non-CS interventions may also be improved in ways that foster the acquisition and mastery of CT skills.

The present study is among the first to demonstrate that children can acquire CT problem-solving skills without necessarily learning to code or engaging in unplugged CS educational activities. The ability of children to acquire CT skills without CS education should not be taken as an argument against the importance of teaching coding at a young age. Developmentally appropriate coding platforms allow children to create personally meaningful projects and can foster positive social and emotional development. Coding provides children with a means of communication and constructive self-expression which are worthwhile ends in themselves. The longitudinal study data presented here and in other studies indicate that learning to code can accelerate the acquisition of CT. Previous work by scholars such as Heckman and Mosso (2014) and Stanovich (1986) has shown that there is a much higher “return of investment” in the long term when interventions target younger populations. For these and other reasons, teaching coding at a young age is advantageous, even if learning to code isn’t the only way to acquire CT skills. This work supports a broader view of computational thinking as a foundational skill set that children acquire through a combination of learning from experience, pedagogical interventions and brain development. It is my hope that this new perspective will help guide researchers, educators, parents and other shareholders as they seek to promote the development of CT skills in young children.

## References

- Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, 55(7), 832–835. <https://doi.org/10.1093/comjnl/bxs074>
- Arfé, B., Vardanega, T., & Ronconi, L. (2020). The effects of coding on children’s planning and inhibition skills. *Computers & Education*, 148, 103807. <https://doi.org/10.1016/j.compedu.2020.103807>
- AVG Technologies research shows number of children aged nine and under able to use an app on a smartphone or tablet increased 38 percent over the last three years (2018). <https://now.avg.com/digital-abilities-overtake-key-development-milestones-for-todays-connected-children>
- Babbage, C. (1832). *On the economy of machinery and manufactures*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1(3), 208–213. <https://doi.org/10.1080/14786443208647876>
- Bakala, E., Gerosa, A., Hourcade, J. P., & Tejera, G. (2021). Preschool children, robots, and computational thinking: A systematic review. *International Journal of Child-Computer Interaction*, 29, 100337. <https://doi.org/10.1016/j.ijcci.2021.100337>
- Barba, L. A. (2016). Computational thinking: I do not think it means what you think it means. <http://lorenabarba.com/blog/computational-thinking-i-do-notthink-it-means-what-you-think-it-means/>.
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48-54. <https://doi.org/10.1145/1929887.1929905>



- Barrouillet, P., & Lecas, J. (1999). Mental models in conditional reasoning and working memory. *Thinking & Reasoning*, 5(4), 289–302. <https://doi.org/10.1080/135467899393940>
- Basu, S., Mustafaraj, E., & Rich, K. (2016). CIRCL primer: Computational thinking. In *CIRCL Primer Series*. <https://circlcenter.org/computationalthinking>
- Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1), 1–48. <https://doi.org/10.18637/jss.v067.i01>
- Bell, T., & Lodi, M. (2019). Constructing computational thinking without using computers. *Constructivist Foundations*, 14(3), 342–351. <https://constructivist.info/14/3/342.bell>
- Bell, T., & Vahrenhold, J. (2018). CS unplugged—How is it used, and does it work? In H.-J. In H.-J. Böckenhauer, D. Komm, & W. Unger (Eds.), *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday*, 497–521. [https://doi.org/10.1007/978-3-319-98355-4\\_29](https://doi.org/10.1007/978-3-319-98355-4_29)
- Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *New Zealand Journal of Applied Computing and Information Technology*, 13, 20–29.
- Bers, M. U. (2008). *Blocks to robots learning with technology in the early childhood classroom*. New York, NY: Teachers College Press.
- Bers, M. U. (2012). *Designing digital experiences for positive youth development: From playpen to playground*. New York, NY: Oxford University Press.
- Bers, M. U. (2017). The Seymour Test: Powerful ideas in early childhood education. *Int. J. Child-Comp. Interact.*, 14(C), 10–14. <https://doi.org/10.1016/j.ijcci.2017.06.004>

- Bers, M. U. (2018). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.
- Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, 6(4), 499–528.  
<https://doi.org/10.1007/s40692-019-00147-3>
- Bers, M. U. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom, Second edition*. Routledge Press.
- Bers, M. U. (2022). *Beyond Coding: How Children Learn Human Values through Programming*. MIT Press.
- Bers M.U., Blake-West J., Govind M., Levinson T., Relkin E., Unahalekhaka A., Yang Z. (2022). *Coding as another language: Research-based curriculum for early childhood computer science* [Manuscript submitted for publication]. Eliot-Pearson Department of Child Study and Human Development, Tufts University.
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72, 145–157. <https://doi.org/10.1016/j.compedu.2013.10.020>
- Bortz, W. W., Gautam, A., Tatar, D., & Lipscomb, K. (2019). The availability of pedagogical responses and the integration of computational thinking. In R. M. Reardon & J. Leonard (Eds.), *Integrating Digital Technology in Education School-University-Community Collaboration* (pp. 81–109). Information Age Publishing-Iap.  
<https://doi.org/10.1007/s10956-019-09805-8>

- Bosgoed L., & Fanchamps, N. (2022). The effect of unplugged programming and visual programming on computational thinking in children aged 5 to 7. *CTE-STEM 2022 Conference*. <https://doi.org/10.34641/ctestem.2022.451>
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. In *Proceedings of the 2012 Annual American Educational Research Association Meeting*. <https://doi.org/10.1.1.296.6602>
- Cetin, I., & Dubinsky, E. (2017). Reflective abstraction in computational thinking. *Journal of Mathematical Behavior*, 47, 70–80. <https://doi.org/10.1016/j.jmathb.2017.06.004>
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers in Education*, 109, 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>.
- Chrysikou, E. G., Caulfield, M. D., & Kan, I. P. (2022). Large-scale network connectivity as a predictor of age: Evidence across the adult lifespan from the Cam-CAN data set. *Psychology and Aging*. Advance online publication. <https://doi.org/10.1037/pag0000683>
- CSTA & ISTE. (2011). *Computational Thinking in K-12 Education: Teacher resources* (Version 2, 2nd ed.). Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE). [https://id.iste.org/docs/ct-documents/ct-teacher-resources\\_2ed-pdf.pdf?sfvrsn=2](https://id.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2)
- Computer Science Teachers Association. (2017). *CSTA K-12 Computer Science Standards, Revised 2017*. Computer Science Teachers Association.

- Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. [Unpublished manuscript].  
<http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>
- Dagienė, V., & Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In R. T. Mittermeir & M. M. Sysło (Eds.), *Informatics Education—Supporting Computational Thinking* (Vol. 5090, pp. 19–30). Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-540-69924-8\\_2](https://doi.org/10.1007/978-3-540-69924-8_2)
- Dagienė, V., & Stupurienė, G. (2016). Bebras--a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in education, 15*(1), 25–44. <https://doi.org/10.15388/infedu.2016.02>.
- Denning, P. J. (2007). Computing is a Natural Science. *Commun. ACM, 50*(7), 13–18.  
<https://doi.org/10.1145/1272516.1272529>
- Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. *Commun. ACM, 60*(6), 33–39. <https://doi.org/10.1145/2998438>
- Denniston, C., Roome, B. R., & Wilson, N. (2015). Efficacy Study of The Foos Game. [White Paper]. KnowProgress.  
<https://edcuration.com/resource/vendor/348/The%20Foos%20Efficacy%20Study%20White%20Paper%20-%20KnowProgress.pdf>
- Diamantopoulos, A., Sarstedt, M., Fuchs, C., Wilczynski, P., & Kaiser, S. (2012). Guidelines for choosing between multi-item and single-item scales for construct measurement: A predictive validity perspective. *Journal of the Academy of Marketing Science, 40*(3), 434–449. <https://doi.org/10.1007/s11747-011-0300-3>

Dijkstra, E. (1979). My hopes for computing science.

<https://www.cs.utexas.edu/users/EWD/transcriptions/EWD07xx/EWD709.html>

diSessa, A. A. (2000). *Changing Minds: Computers, Learning, and Literacy*. MIT Press.

diSessa, A. A. (2018). Computational Literacy and “The Big Picture” Concerning Computers in

Mathematics Education. *Mathematical Thinking and Learning*, 20(1), pp. 3-31,

<https://doi.org/10.1080/10986065.2018.1403544>

Dorn, E., Hancock, B., Sarakatsannis, J., & Viruleg, E. (2020). COVID-19 and student learning in the United States: The hurt could last a lifetime. McKinsey & Company, 1.

El-Hamamsy, L., Zapata-Cáceres, M., Barroso, E. M., Mondada, F., Zufferey, J. D., & Bruno, B.

(2022). The competent Computational Thinking Test: Development and Validation of an

Unplugged Computational Thinking Test for Upper Primary School. *Journal of*

*Educational Computing Research*, 07356331221081753.

<https://doi.org/10.1177/07356331221081753>

*etymonline*. (n.d.). *Computational | Etymology, origin and meaning of computational by*

<https://www.etymonline.com/word/computational>

Ezeamuzie, N., & Leung, J. (2021). Computational Thinking Through an Empirical Lens: A

Systematic Review of Literature. *Journal of Educational Computing Research*,

073563312110331. <https://doi.org/10.1177/07356331211033158>

Fias, W., Menon, V., & Szucs, D. (2013). Multiple components of developmental dyscalculia.

*Trends in Neuroscience and Education*, 2(2), 43–47.

<https://doi.org/10.1016/j.tine.2013.06.006>

Field, A. (2009) *Discovering Statistics Using SPSS*. 3rd Edition, Sage Publications Ltd., London.

- Flavell, J. H., Miller, P. H., & Miller, S. A. (1993). *Cognitive development (3rd ed.)*. Prentice Hall. NJ.
- Frailon, J., Ainley, J., Schulz, W., Friedman, T., & Duckworth, D. (2018). Preparing for life in a digital world: IEA International computer and information literacy study 2018 international report (Vol. 297). Springer Nature.  
<https://doi.org/10.1007/978-3-030-38781-5>
- Gerdes, J., Durden, T., & Poppe, L. (2013). Brain Development and Learning in the Primary Years .G2198. *Faculty Publications from Nebraska Center for Research on Children, Youth, Families, and Schools*. <https://digitalcommons.unl.edu/cyfsfacpub/77>
- Gerosa, A., Koleszar, V., Gonzalo, T., Leonel, G.-S., & Alejandra, C. (2021). Cognitive abilities and computational thinking at age 5: Evidence for associations to sequencing and symbolic number comparison. *Computers and Education Open*, 2, 100043.  
<https://doi.org/10.1016/j.caeo.2021.100043>
- Gerson, S. A., Morey, R. D., & van Schaik, J. E. (2022). Coding in the cot? Factors influencing 0–17s’ experiences with technology and coding in the United Kingdom. *Computers & Education*, 178, 104400. <https://doi.org/10.1016/j.compedu.2021.104400>
- Google. (n.d.). Exploring Computational Thinking. <http://g.co/exploringct> – The page has now been removed, but can be found in the “CT overview” tab here:  
<https://web.archive.org/web/20181001115843/https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview>
- Govind, M., Relkin, E., & Bers, M. U. (2020). Engaging Children and Parents to Code Together Using the ScratchJr App. *Visitor Studies*. <https://doi.org/10.1080/10645578.2020.1732184>

- Grillo-Hill, A., Mahoney, C., Chow, E., & Li, L. (2019). StoryCoder Classroom Feasibility Study [White Paper]. WestEd.  
[https://edcuration.com/resource/vendor/348/codeSpark\\_Feasibility%20Memo\\_Draft.pdf](https://edcuration.com/resource/vendor/348/codeSpark_Feasibility%20Memo_Draft.pdf)
- 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>
- Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 57–62). *ACM*. <https://doi.org/10.1145/2591708.2591713>.
- Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2010). *Multivariate data analysis* (7th ed.). Englewood Cliffs: Prentice Hall
- Hartle, L. C. (2019). Technology and Young Children: Processes, Context, Research, and Practice. In L.E. Cohen & S. Waite-Stupiansky (Eds.), *STEM for Early Childhood Learners: How Science, Technology, Engineering and Mathematics Strengthen Learning* (pp. 22-45). New York, NY: Routledge.
- Hassenfeld, Z. R., Govind, M., de Ruiter, L. E., & Bers, M. U. (2020). If You Can Program, You Can Write: Learning Introductory Programming Across Literacy Levels. *Journal of Information Technology Education: Research*, 19, 65-85. <https://doi.org/10.28945/4509>
- Heckman, J. J., & Mosso, S. (2014). The Economics of Human Development and Social Mobility. *Annual review of economics*, 6, 689–733.  
<https://doi.org/10.1146/annurev-economics-080213-040753>
- Hermans, F., & Aivaloglou, E. (2017). To Scratch or Not to Scratch? A Controlled Experiment Comparing Plugged First and Unplugged First Programming Lessons. *Proceedings of the*

*12th Workshop on Primary and Secondary Computing Education*, 49–56.

<https://doi.org/10.1145/3137065.3137072>

Hinton, P., Brownlow, C., McMurray, I., & Cozens, B. (2004). *SPSS explained*.

Abingdon-on-Thames: Taylor & Francis. <https://doi.org/10.4324/9780203642597>.

Hogenboom, S. A. M., Hermans, F. F. J., & Maas, H. L. J. V. der. (2021). Computerized adaptive assessment of understanding of programming concepts in primary school children.

*Computer Science Education*, 0(0), 1–30.

<https://doi.org/10.1080/08993408.2021.1914461>

Horn, M. (2012). *TopCode: Tangible Object Placement Codes*.

<http://users.eecs.northwestern.edu/~mhorn/topcodes>.

Iseli, M. R., Feng, T., Relkin, E., & Chung, G. K. W. K. (2021a). Evaluation of Code

Manipulation in Coding Games. *Virtual Annual Meeting of the American Educational Research Association (AERA)*.

Iseli, M. R., Feng, T., Chung, G. K. W. K., Ruan, Z., Shochet, J., & Stachman, A. (2021b).

Using Visualizations of Students' Coding Processes to Detect Patterns Related to Computational Thinking. Paper presented at 2021 ASEE Virtual Annual Conference Content Access, Virtual Conference. <https://peer.asee.org/38006>.

Iseli, M. R., Relkin, E., Zhang, Y., Chung, G. K. W. K., Shochet, J., Strachman, A., Hosford, G., (2022). *Defining Computational Thinking Using Semantic Analysis of Prior Definitions*.

[Manuscript in Preparation]. CRESST, UCLA

Janveau-Brennan, G., & Markovits, H. (1999). The development of reasoning with causal conditionals. *Developmental Psychology*, 35(4), 904–911.



- Jenson, J., & Droumeva, M. (2016). Exploring media literacy and computational thinking: A game maker curriculum study. *Electronic Journal of e-Learning*, 14(2), 111-121.  
<https://academic-publishing.org/index.php/ejel/article/view/1748>
- Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review.  
[https://www.researchgate.net/publication/303943002\\_A\\_Framework\\_for\\_Computational\\_Thinking\\_Based\\_on\\_a\\_Systematic\\_Research\\_Review](https://www.researchgate.net/publication/303943002_A_Framework_for_Computational_Thinking_Based_on_a_Systematic_Research_Review)
- Kanaki, K., & Kalogiannakis, M. (2022). Assessing Algorithmic Thinking Skills in Relation to Gender in Early Childhood. *Educational Process International Journal*, 11, 44–59.  
<https://doi.org/10.22521/edupij.2022.112>.
- Katz, D. L. (1960). Conference report on the use of computers in engineering classroom instruction. *Communications of the ACM*, 3(10), 522–527.  
<https://doi.org/10.1145/367415.993453>
- Kingsbury, G. G., & Weiss, D. J. (1983). A comparison of IRT-based adaptive mastery testing and a sequential mastery testing procedure. In *New horizons in testing* (pp. 257-283). Academic Press. <https://doi.org/10.1016/B978-0-12-742780-5.50024-X>.
- Knuth, D. E. (1974). Computer science and its relation to mathematics. *The American Mathematical Monthly*, 81(4), 323-343.  
<https://doi.org/10.1080/00029890.1974.11993556>
- Ladner, R. E., & Stefik, A. (2017). AccessCSforall: Making Computer Science Accessible to K-12 Students in the United States. *SIGACCESS Access. Comput.*, 118, 3–8.  
<https://doi.org/10.1145/3124144.3124145>

- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32-37. doi: <https://doi.org/10.1145/1929887.1929902>
- Levinson, T., & Bers, M. U. (2022). *Student Centered Computational Thinking for Children with Disabilities*. American Educational Research Association (AERA) Annual Meeting, San Diego, CA.
- Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020). Computational Thinking Is More about Thinking than Computing. *Journal for STEM Education Research*, 3(1), 1–18. <https://doi.org/10.1007/s41979-020-00030-2>
- Lockwood, J., & Mooney, A. (2018). Computational Thinking in education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools*, 2(1), 41-60. <https://doi.org/10.21585/ijcses.v2i1.26>
- Lodi, M. (2020). Informatical Thinking. *Olympiads in Informatics: An International Journal*, Vilnius University, International Olympiad in Informatics, 2020, 14, pp.113-132. <https://doi.org/10.15388/ioi.2020.09 . hal- 02981734>
- Lodi, M., & Martini, S. (2021). Computational Thinking, Between Papert and Wing. *Science & Education*, 30(4), 883–908. <https://doi.org/10.1007/s11191-021-00202-5>
- Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2017). *Stuck in the shallow end: Education, race, and computing*. MIT Press. <https://mitpress.mit.edu/books/stuck-shallow-end>
- Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the cognition of coding in 3-to-6-year olds: the development of an assessment tool and the

- relation between coding ability and cognitive compiling of syntax in natural language. Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER '18, 133–141. <https://doi.org/10.1145/3230977.3230984>.
- Metin, S. (2020). Activity-based unplugged coding during the preschool period. *International Journal of Technology and Design Education*, 32(1), 149–165. <https://doi.org/10.1007/s10798-020-09616-8>
- Mioduser, D., & Levy, S. T. (2010). Making Sense by Building Sense: Kindergarten Children's Construction and Understanding of Adaptive Robot Behaviors. *International Journal of Computers for Mathematical Learning*, 15(2), 99–127. <https://eric.ed.gov/?id=EJ924252>
- Mioduser, D., Levy, S. T., & Talis, V. (2009). Episodes to scripts to rules: Concrete-abstractions in kindergarten children's explanations of a robot's behavior. *International Journal of Technology and Design Education*, 19(1), 15–36. <https://doi.org/10.1007/s10798-007-9040-6>
- Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple representations in computational thinking tasks: a clinical study of second-grade students. *Journal of Science Education and Technology*, 29(1), 19–34. <https://doi.org/10.1007/s10956-020-09812-0>.
- Muller, U., Overton, W. F., & Reene, K. (2001). Development of conditional reasoning: A longitudinal study. *Journal of Cognition and Development*, 2(1), 27–49.
- National Research Council. (2010). Report of a workshop on the scope and nature of computational thinking. Washington, DC: National Academies Press.
- National Research Council. (2011). Report of a workshop on the pedagogical aspects of computational thinking. *National Academies Press*.

- Odegard, N. (2012). When matter comes to matter – working pedagogically with junk materials. *Education Inquiry*, 3(3), p.387-400.
- Organisation for Economic Co-operation and Development (OECD), (2010). What do we know about children and technology? *Educational Research and Innovation, OECD Publishing*, Paris. <https://www.oecd.org/education/ceri/Booklet-21st-century-children>
- Peters, L., & Smedt, B. D. (2018). Arithmetic in the developing brain: A review of brain imaging studies. *Developmental Cognitive Neuroscience*, 30, 265–279.  
<https://doi.org/10.1016/j.dcn.2017.05.002>
- Papavlasopoulou, S., Sharma, K., & Giannakos, M. N. (2020). Coding activities for children: Coupling eye-tracking with qualitative data to investigate gender differences. *Computers in Human Behavior*, 105, Article 105939. <https://doi.org/10.1016/j.chb.2019.03.003>
- Papert, S. (1993). *The children's machine: Rethinking schools in the age of the computer*. New York: Basic Books.
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Papert, S. (1996). An Exploration in the Space of Mathematics Educations, *International Journal of Computers for Mathematical Learning*, Vol. 1, No. 1, pp. 95-123.
- Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal*, 39(3.4), 720–729. <https://doi.org/10.1147/sj.393.0720>
- Papert, S., & Harel, I. (1991). *Situating Constructionism*. In *Constructionism*. Ablex Publishing Corp.
- Penfold, L. (2019). Material Matters in Children's Creative Learning. *Journal of Design and Science*. <https://jods.mitpress.mit.edu/pub/bwp6cysy>

- Perlis, A. J. (1963). The computer in the university. In M. Greenberger, Ed., *Computers and the World of the Future*, MIT Press, Cambridge, MA, 180–219.
- Piaget, J. (1971). Developmental stages and developmental processes. In D. R. Green, M. P. Ford, & G. B. Flamer (Eds.), *Measurement and Piaget* (pp. 172–188). New York: McGraw-Hill.
- Portelance, D. J., & Bers, M. U. (2015). Code and tell: Assessing young children’s learning of computational thinking using peer video interviews with ScratchJr. *Proceedings of the 14th International Conference on Interaction Design and Children - IDC ’15*, 271–274. <https://doi.org/10.1145/2771839.2771894>
- Prakken, L. W. (1942). *The Education Digest* Vol 8 Page 49
- R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>
- Relkin E. & Bers, M. (2021). *TechCheck-K: A Measure of Computational Thinking for Kindergarten Children*. In *2021 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. <https://sites.tufts.edu/devtech/files/2021/05/1487.pdf>
- Relkin E., de Ruiter., L., Bers, M.U. (2021). Learning to Code and the Acquisition of Computational Thinking by Young Children. *Computers & Education*. <https://doi.org/10.1016/j.compedu.2021.104222>
- Relkin, E. (2018). Assessing young children’s computational thinking abilities (Master’s thesis). Retrieved from ProQuest Dissertations and Theses database. (UMI No. 10813994).
- Relkin, E. (2021). Creation of an unplugged computational thinking assessment for young children. In M. U. Bers (Ed.) *Teaching Computational Thinking and Coding to Young Children* (pp. 250-264). IGI Global. <https://doi.org/10.4018/978-1-7998-7308-2.ch013>

Relkin, E. & Bers, M. U. (2020). Exploring the Relationship Among Coding, Computational Thinking, and Problem Solving in Early Elementary School Students [Symposium]. *Annual Meeting of the American Educational Research Association (AERA)*, San Francisco, CA (Conference Cancelled).

<https://sites.tufts.edu/devtech/files/2021/05/RelkinBersAERA20.pdf>

Relkin, E., & Strawhacker, A. (2021). Unplugged learning: Recognizing computational thinking in everyday life. In M. U. Bers (Ed.) *Teaching Computational Thinking and Coding to Young Children* (pp. 41-62). IGI Global.

<https://doi.org/10.4018/978-1-7998-7308-2.ch003>

Relkin, E., de Ruiter., L., Bers, M.U. (2020). TechCheck: Development and Validation of an Unplugged Assessment of Computational Thinking in Early Childhood Education. *Journal of Science Education and Technology*.

<https://doi.org/10.1007/s10956-020-09831-x>

Resnick, M., & Robinson, K. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. MIT press.

<https://mitpress.mit.edu/books/lifelong-kindergarten>

Román-González, M., Moreno-León, J., Robles, G. (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In: Kong, SC., Abelson, H. (eds) *Computational Thinking Education*. Springer, Singapore.

[https://doi.org/10.1007/978-981-13-6528-7\\_6](https://doi.org/10.1007/978-981-13-6528-7_6)

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational

- Thinking Test. *Computers in Human Behavior*, 72, 678–691.  
<https://doi.org/10.1016/j.chb.2016.08.047>
- Román-González, M., Pérez-González, J.-C., Moreno-León, J., & Robles, G. (2018). Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior*, 80, 441–459. <https://doi.org/10.1016/j.chb.2017.09.030>
- Romano, J., Kromrey, J., Coraggio, J. & Skowronek, J. (2006). Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys?. In *annual meeting of the Florida Association of Institutional Research* (pp. 1-3) .
- Santos, J. S., Andrade, W. L., Brunet, J., & Araujo Melo, M. R. (2020). A Systematic Literature Review of Methodology of Learning Evaluation Based on Item Response Theory in the Context of Programming Teaching. *2020 IEEE Frontiers in Education Conference (FIE)*, 1–9. <https://doi.org/10.1109/FIE44824.2020.9274068>
- Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition* [Monograph]. <https://eprints.soton.ac.uk/356481/>
- Shute, V., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. <https://doi.org/10.1016/j.edurev.2017.09.003>
- Stanovich, K. E. (1986). Matthew effects in reading: Some consequences of individual differences in the acquisition of literacy. *Reading Research Quarterly*, 21(4), 360-407.
- Strawhacker, A. L., Lee, M. S. C., & Bers, M. U. (2017). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*.  
<https://doi.org/10.1007/s10798-017-9400-9>

- Strawhacker, A., & Bers, M. U. (2019). What They Learn When They Learn Coding: Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development*, 67(3), 541-575.  
<https://doi.org/10.1007/s11423-018-9622-x>
- Sullivan, A. (2016). Breaking the STEM Stereotype: Investigating the Use of Robotics to Change Young Children's Gender Stereotypes About Technology & Engineering. ProQuest Dissertations and Theses database. <http://hdl.handle.net/10427/011851>
- Sullivan, A. & Bers, M.U. (2017). Computational Thinking and Young Children: Understanding the Potential of Tangible and Graphical Interfaces. In Ozcinar, H., Wong, G., & Ozturk, T. (Eds.) *Teaching Computational Thinking in Primary Education*. IGI Global.  
<https://doi.org/10.4018/978-1-5225-3200-2.ch007>
- Sullivan, A., & Bers, M. U. (2016). Girls, boys, and bots: Gender differences in young children's performance on robotics and programming tasks. *Journal of Information Technology Education: Innovations in Practice*, 15, 145- 165.  
<http://www.informingscience.org/Publications/3547>
- Sullivan, A., Bers, M. U., Mihm, C. (2017). Imagining, Playing, & Coding with KIBO: Using KIBO Robotics to Foster Computational Thinking in Young Children. *Proceedings of the International Conference on Computational Thinking Education*. Wanchai, Hong Kong.  
<https://www.eduhk.hk/cte2017/doc/CTE2017%20Proceedings.pdf#page=121>
- Swade, D. D. (2005). The Construction of Charles Babbage's Difference Engine No. 2. *IEEE Annals of the History of Computing*, 27(3), 70–78.  
<https://doi.org/10.1109/MAHC.2005.45>



- Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, 103798. <https://doi.org/10.1016/j.compedu.2019.103798>
- The Mathematics Teacher (1943). National Council of Teachers of Mathematics Vol 36 Issue 2
- Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education, 162*, 104083. <https://doi.org/10.1016/j.compedu.2020.104083>
- Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research, 57*(1), 3–31. <https://doi.org/10.1177/0735633117743918>
- Hançer, N., Çiftçi, A., & Topcu, M. (2021). *Turkish Early Childhood Children's Computational Thinking Skills: Adaptation of TechCheck-K to Turkish*. In *The International Conference on Science and Education*, Antalya, Turkey. [https://www.isres.org/conferences/2021\\_Antalya/ICONSE2021\\_Abstract.pdf](https://www.isres.org/conferences/2021_Antalya/ICONSE2021_Abstract.pdf)
- Vogel, S. E., & De Smedt, B. (2021). Developmental brain dynamics of numerical and arithmetic abilities. *NPJ Science of Learning, 6*(1), 22. <https://doi.org/10.1038/s41539-021-00099-3>
- Vizner M. Z. (2017). Big robots for little kids: investigating the role of scale in early childhood robotics kits (Master's thesis). Available from ProQuest Dissertations and Theses database. (UMI No.10622097).
- Wang, J., & Hejazi Moghadam, S. (2017). Diversity Barriers in K-12 Computer Science Education: Structural and Social. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 615–620. <https://doi.org/10.1145/3017680.3017734>

- Wang, D., Wang, T., & Liu, Z. (2014). A tangible programming tool for children to cultivate computational thinking. *Scientific World Journal*, 428080.  
<https://doi.org/10.1155/2014/428080>
- Wang, C., Chao, J., & Shen, J. (2021). Integrating Computational Thinking in STEM Education: A Literature Review. *International Journal of Science and Mathematics Education*. <https://doi.org/10.1007/s10763-021-10227-5>
- Wang, J., & Hejazi Moghadam, S. (2017). Diversity barriers in K-12 computer science education: structural and social. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 615-620.
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147.  
<https://doi.org/10.1007/s10956-015-9581-5>
- Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. *Learning, Education And Games*, 37.  
<https://dl.acm.org/citation.cfm?id=2811150>.
- Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220.  
<https://doi.org/10.1145/2157136.2157200>
- Wickham, H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.  
<https://ggplot2.tidyverse.org>

- Williams, D. R., Priest, N., & Anderson, N. B. (2016). Understanding associations among race, socioeconomic status, and health: Patterns and prospects. *Health psychology : official journal of the Division of Health Psychology, American Psychological Association*, 35(4), 407–411. <https://doi.org/10.1037/hea0000242>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3),33–35. <https://doi.org/10.1145/1118178.1118215>
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717-3725. <https://doi.org/10.1098/rsta.2008.0118>
- Wing, J. M. (2011). Research notebook: Computational thinking—What and why? *The Link Magazine*, Spring. Carnegie Mellon University, Pittsburgh. <https://www.cs.cmu.edu/link/research-notebookcomputational-thinking-what-and-why>
- Wohl, B., Porter, B., & Clinch, S. (2015). Teaching computer science to 5–7 yearolds: An initial study with scratch, cubelets and unplugged computing. *Proceedings of the Workshop in Primary and Secondary Computing Education*, 55–60. 10.1145/2818314.2818340
- Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017a). Computational Thinking as an Emerging Competence Domain. In M. Mulder (Ed.), *Competence-based Vocational and Professional Education: Bridging the Worlds of Work and Education* (pp. 1051–1067). Springer International Publishing. [https://doi.org/10.1007/978-3-319-41713-4\\_4](https://doi.org/10.1007/978-3-319-41713-4_4)
- Yadav, A., Gretter, S., Good, J., & McLean, T. (2017b). Computational Thinking in Teacher Education. In P. J. Rich & C. B. Hodges (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 205–220). Springer International Publishing. [https://doi.org/10.1007/978-3-319-52691-1\\_13](https://doi.org/10.1007/978-3-319-52691-1_13)

- 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), 5. <https://doi.org/10.1145/2576872>
- Yang, W., Ng, D. T. K., & Gao, H. (2021). Robot programming versus block play in early childhood education: Effects on computational thinking, sequencing ability, and self-regulation. *British Journal of Educational Technology*.  
<https://doi.org/10.1111/bjet.13215>
- Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2020). Computational thinking test for Beginners: Design and content validation. In 2020 IEEE global engineering education conference (EDUCON) (pp. 1905–1914). *IEEE*.  
<https://doi.org/10.1109/EDUCON45650.2020.9125368>.
- Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, 141, 103607.  
<https://doi.org/10.1016/j.compedu.2019.103607>.



# Learning to code and the acquisition of computational thinking by young children

E. Relkin<sup>\*</sup>, L.E. de Ruiter, M.U. Bers

Eliot-Pearson Department of Child Study and Human Development, Tufts University, USA

## ARTICLE INFO

### Keywords:

Computational thinking  
Coding  
Early childhood education  
Unplugged assessment  
Curriculum

## ABSTRACT

This longitudinal study examined changes in Computational Thinking (CT) skills in first and second grade students exposed to a developmentally appropriate coding curriculum. The “Coding as Another Language” (CAL) curriculum spans seven weeks and uses the KIBO robot to engage students in learning that integrates programming and literacy concepts. We compared children receiving CAL ( $N = 667$ ) to a control group ( $N = 181$ ) who participated in typical classroom activities without coding (No-CAL). *TechCheck*, a validated “unplugged” CT assessment suitable for young children regardless of their coding experience, was used to measure CT. Over the course of the study, children who received CAL-KIBO improved on *TechCheck* ( $M_{change} = 0.94, p < .001$ ) whereas the No-CAL group did not change significantly ( $M_{change} = 0.27, p = .07$ ). Accounting for demographic factors, baseline performance and classroom (teacher) effects, CAL exposure was a significant predictor of post-test CT scores ( $p < .01$ ). Improvements in CT measured by *TechCheck* over seven weeks of the CAL-KIBO curriculum were consistent with approximately six months of development without coding instruction. Secondary analysis stratified by grade revealed decisive evidence that CAL exposure improved scores in first grade and anecdotal evidence that second grade scores improved. The CT domains that showed improvement in children who received CAL-KIBO included algorithms, modularity, and representation. Young children who learned to code improved in solving unplugged problems that were not explicitly taught in the coding curriculum. This provides evidence that a developmentally appropriate curriculum for teaching young children to code can accelerate their acquisition of CT skills.

## 1. Introduction

One of the most important goals of teaching computer science (CS) to young children is to foster the development of computational thinking (CT) skills that are applicable to many educational disciplines and areas of life (Barr & Stephenson, 2011; Chen et al., 2017; Cuny et al., 2010; Wing, 2006). Papert (1980) alluded to CT in his book *Mindstorms* in a discussion of the challenge of integrating Computer Science (CS) education with children’s everyday experiences. Later, Wing popularized the term and defined it as a set of reasoning skills for formulating and solving problems using computers and other information technologies (Wing, 2006, 2011). She emphasized that CT is not only useful in CS but also other disciplines such as mathematics, science, design, economics, and linguistics (Wing, 2011). Since that time there has been increasing interest in CT, as documented in several recent reviews describing CT’s definitions, methods of assessment and educational initiatives (Lye & Koh, 2014; Román-González et al., 2019; Tang et al., 2020;

<sup>\*</sup> Corresponding author. Eliot-Pearson Department, 105 College Ave, Medford, MA, 02155, USA.  
E-mail address: [Emily.relkin@tufts.edu](mailto:Emily.relkin@tufts.edu) (E. Relkin).

Zhang & Nouri, 2019).

There is an ongoing debate about whether CT is truly a singular concept (Barr et al., 2011; Grover & Pea, 2013; National Research Council, 2011). Zhang and Nouri (2019) identified three types of definitions of CT in the published literature: *generic definitions* that focus on universal problem-solving skills (e.g., Aho, 2012; Wing, 2011); *operational definitions* that provide a vocabulary and identify CT sub-domains (e.g., CSTA, 2011; Selby & Woollard, 2013) and *educational definitions* that provide concepts and competencies (e.g., Barr & Stephenson, 2011; Brennan & Resnick, 2012). Tang et al. (2020) later distinguished CT definitions that are programming-related (e.g., Grover, et al., 2015) from those of a general problem-solving nature (e.g., CSTA, 2011; Selby & Woollard, 2013). The existence of many different definitions is an indication that CT is still an evolving concept but one recognized to have considerable importance for CS education. For present purposes, we define CT to be a set of heuristic reasoning skills that can be categorized into discrete sub-domains applicable to problem-solving in computer science and other disciplines.

Coding (programming) has been described as “the instrumental skill of CT” and “the primary means of teaching CT in primary school” (Arfé et al., 2019; Román-González, 2017; Wing, 2006). Programming languages are specifically designed to communicate instructions and solve problems with computers, and children as young as 3–4 years of age are capable of learning to code (Bers, 2018; Clements & Gullo, 1984; Kazakoff & Bers, 2014; Strawhacker & Bers, 2019). However, in a 2014 review Lye and Koh (2014) found that the majority of past studies of coding and CT were carried out in higher educational settings, and only 25% involved kindergarten through 12th grade students. Lockwood and Mooney (2018) conducted a systematic review of CT in secondary schools (children ages 11–18) and concluded that educational programs promoting CT in middle and high schools are becoming more widespread. While there has been an increase in CT educational initiatives and professional development programs for younger students and their teachers (Fraillon et al., 2018; Tang et al., 2020), more work is needed in this area. In particular, there is still only a limited understanding of the effects of learning to code on young children’s cognitive development and how to best promote the development of CT.

### 1.1. Teaching computational thinking to young children

Educational initiatives relating to CT in young children must take into account the progression of cognitive development. A typically developing young child does not possess fully mature literacy, numeracy, and abstract reasoning skills (Piaget, 1971). According to developmental theorists, first and second grade children are typically in the preoperational or concrete operations stage. At the preoperational stage from around two years to six years of age, children tend to engage in concrete, egocentric thinking and are just beginning to develop knowledge about physical symbols and representation. By the concrete operations stage from approximately six to twelve years, they are better able to organize their thoughts, use logical reasoning skills, and rely less directly on physical representations of ideas (Bruner et al., 1966; Feldman, 2004; McDevitt & Ormrod, 2002; Piaget, 1953).

A young child’s stage of development can constrain the CS concepts and CT skills they can readily master (Chen et al., 2017; Goldstein & Flake, 2016). For example, early elementary school children may have difficulty grasping “if-then” conditionals (Barrouillet & Lecas, 1999; Janveau-Brennan & Markovits, 1999; Muller et al., 2001). Likewise, they may have a hard time understanding abstract representations such as variables. They may engage in magical thinking or personification rather than recognize the mechanical basis for the actions of machines (Flavell et al., 1993; Mioduser et al., 2009). These and other developmental considerations must be taken into account when designing educational programs to teach CT to young children.

In an effort to provide a developmentally appropriate framework for teaching coding and other CS concepts to children between the ages of 4–9 years, Bers (2018) described the seven powerful ideas of CS. This framework is based on experience with a variety of coding initiatives for children, such as Google for Education, 2010; Scratch (Brennan & Resnick, 2012); the KIBO robotics kit (Sullivan & Bers, 2015) and ScratchJr (Portelance et al., 2015). The seven powerful ideas identify child-friendly concepts within the domains of hardware/software, algorithms, modularity, control structures, representation, debugging, and design process (see Table 1).

The powerful ideas provided the foundation for the CS curriculum used in the current study called “Coding as Another Language” (CAL). This curriculum is designed to teach coding and CT to young children while simultaneously promoting literacy skills (Bers, 2018; Hassenfeld et al., 2020). Programming in elementary education has typically been associated with Science, Technology, Engineering and Mathematics (STEM) curricula (Bers, 2019; Clements et al., 2001; Guzdial & Morrison, 2016). However, there are

**Table 1**

The seven powerful ideas, associated concepts, and examples from the CAL-KIBO curriculum.

Powerful Idea	Associated Concepts	Example from CAL-KIBO Curriculum
Algorithms	Sequencing/order, logical organization	Child learns to program KIBO in a specific sequence to dance the “Hokey Pokey”
Modularity	Breaking up larger task into smaller parts, instructions	Students break up the “If You’re Wild and You Know It” song into smaller components that KIBO can be programmed to perform
Control Structures	Recognizing patterns and repetition, cause and effect	Children learn to trigger sound sensors using “wait for clap” command
Representation	symbolic representation, models	Child learns that each programming block translates into a unique KIBO action.
Hardware/Software	Smart objects are not magical, objects are human engineered	Children play a game about what is and isn’t a robot and learn that you must give the KIBO robot a program in order for it to perform
Design Process	Problem solving, perseverance, editing/revision	Children are tasked with creating a final “Wild Rumpus” KIBO project in which they plan, code, test and revise with peer sharing and feedback
Debugging	Identifying problems, problem solving, perseverance	Children identify problems in either hardware or software of KIBO and brainstorm solutions to fix it

creative and self-expressive aspects of programming that align more closely with literacy and other aspects of the humanities (Bers, 2018, 2020; Resnick & Siegel, 2015). The CAL curriculum draws on principles of literacy education to create lessons that blend elements of learning to read and write with CS and coding concepts (Bers, 2019; Hassenfeld et al., 2020). This pedagogical approach emphasizes creative programming and provides children with opportunities for self-expression analogous to those experienced when using a symbolic written language (Bers, 2018, 2019).

One of the greatest challenges to integrating CT into early elementary school education has been a lack of validated, developmentally appropriate assessments to measure young children's CT skills in classroom and online settings (Lockwood & Mooney, 2018; Lee et al., 2011; Román-González et al., 2019). In the following section, we review the development of CT assessments for young children and describe the recent advent of “unplugged” CT assessments such as *TechCheck*, the instrument employed in this study.

### 1.2. Assessing computational thinking in young children

CT assessment instruments for young children must use developmentally appropriate language and tasks to assure that factors such as literacy and fine motor skills are not limiting (Chen et al., 2017; Sattler, 2014). Cultural biases should be avoided, and the activities and artifacts employed must be familiar and non-threatening to young children (McMillan, 2013; Mullis & Martin, 2019; Tang et al., 2020). The duration of the assessment should be relatively brief in light of the shorter attention span of young children (Moyer & Gilmer, 1953). The range of difficulties covered by the assessment should allow for children with little or no CT training to be assessed with equal ease and precision to students with extensive CT talent (Relkin et al., 2020). It has been suggested that CT assessments should incorporate measures that evaluate reasoning processes, not just the end product of a program or a problem solved (Brennan & Resnick, 2012; Fields et al., 2019; Román-González et al., 2019). However, this is arguably an aspirational goal that has yet to be achieved in a brief CT assessment that can be administered to large numbers of young students simultaneously in a classroom setting.

Román-González et al. (2019) reviewed CT assessment tools for kindergarten through 12th grade and found most were designed for students in middle school, high school and/or adults (Chen et al., 2017; Fraillon et al., 2018; Román-González et al., 2018; Werner et al., 2012). Some CT assessments require hours and/or multiple sessions to complete, making them impractical for routine use in educational settings (Basu et al., 2016; Chen et al., 2017; Werner et al., 2014). CT assessments that employ programming challenges that require some prior knowledge of coding may conflate programming abilities with CT skills (Yadav et al., 2017). Such instruments cannot readily be used to assess baseline CT abilities in coding-naïve students. To the extent that it is desirable to be able to measure CT skills in children regardless of whether they have past knowledge or experience with computer programming, coding exercises alone may not be the best way to assess CT (Grover et al., 2014).

Recently, our research group and others have explored the use of coding-free instruments to assess CT skills in children. These newer instruments leverage the fact that CT skills can be exercised without programming through the use of unplugged activities (Bell & Vahrenhold, 2018; Zapata-Cáceres et al., 2020). Unplugged activities consist of puzzles, games and other exercises that draw upon CS concepts without requiring explicit knowledge of coding or computers. Unplugged activities have been used to teach CS concepts for over two decades (e.g., CSUnplugged.com; code.org) and can also be used for assessment purposes.

One of the first unplugged CT assessments was designed for post-elementary school students by Román-González et al. (2018) who created a 45-min unplugged assessment called the Computational Thinking Test (CTt). This instrument was used to measure CT abilities in over 300 middle school students (ages 12–14) before and after they took part in an informatics course that included elements of the code.org curriculum (Román-González et al., 2018). After the coding intervention, CTt assessment scores improved and correlated positively ( $p < .01$ ) with language ( $r = 0.42$ ), grade point average ( $r = 0.47$ ), mathematics ( $r = 0.36$ ) and informatics ( $r = 0.43$ ). In its original form, the CTt is not suitable for use in younger, elementary school-age children.

Arfé et al. (2019) used four traditional neuropsychological tests to measure executive functioning in first and second graders who received components of the code.org curriculum. The tests did not involve coding or computer technology and as such could be considered “unplugged.” Among  $n = 42$  first graders who received 8 h of coding instruction, the measures of response inhibition and planning improved more than in a control group ( $n = 34$ ) that received non-coding STEM activities (Arfé et al., 2019). The authors also followed  $n = 17$  second grade students longitudinally and found that changes in planning and response inhibition after one month of

**Table 2**

Comparison of Two “Unplugged” CT assessments for young children: The BCTt and *TechCheck* based on Zapata-Cáceres et al., 2020 and Relkin et al., 2020.

	BCTt	<i>TechCheck</i>
Average Admin Time	40 min	13 min
Format	Pen and paper	Pen and paper, Online
Validation Sample	299 students	768 students
Validated Age Range	5-12 (1st- 6th grade)	5-9 (1st - 2nd grade)
Age Sensitivity	Significant difference between 2nd grade vs. 4th and 6th graders in initial validation study. No significant difference between 1st and 2nd graders reported. No difference between 4th and 6th graders	Significant difference between 1st and 2nd graders. No data on older or younger children in initial validation study.
CT Concepts	Sequences, Loops (Simple, Nested), Conditionals (If-Then, If-Then-Else, While)	Algorithms, Modularity, Debugging, Hardware/Software, Control Structures, Representation



the coding intervention were comparable to those that occurred over seven months of normal development. This study provides evidence from a randomized, control trial that learning to code can accelerate the development of executive functions critical to CT in young children. However, the number of participants was relatively small and the assessment measures employed focused on a specific subset of the various skills involved in CT. In light of this, further studies are needed to evaluate the impact of learning to code on young children's CT skills.

Cross-sectional validation studies were recently completed on two unplugged CT assessments designed specifically for young children. The *CTt for Beginners* (BCTt) (Zapata-Cáceres et al., 2020) and *TechCheck* (Relkin et al., 2020) both use unplugged challenges to probe CT domains and can be administered to children who lack prior coding experience. These instruments differ in the types of unplugged challenges they include, the CT domains assessed, the targeted age ranges and the time required to complete the assessments (see Table 2).

Although both the BCTt and *TechCheck* assessment instruments have unique merits, *TechCheck* was chosen for the present study for several reasons. *TechCheck*'s CT constructs are based on Bers' seven powerful ideas, the same conceptual foundation as the CAL coding curriculum used in this study. On average, *TechCheck* takes approximately 13 min to administer while the BCTt requires approximately 40 min. Some of the concepts probed by the BCTt such as conditionals may be problematic for younger children on developmental grounds (Barrouillet & Lecas, 1999; Janveau-Brennan & Markovits, 1999; Muller et al., 2001). Mean scores on *TechCheck* were significantly different in first and second graders whereas no significant difference between these grades was reported for the BCTt (Zapata-Cáceres et al., 2020). In addition, *TechCheck* can be administered to large groups of children simultaneously using an online platform, which is useful in the context of the present study involving hundreds of students.

By obtaining a better understanding of how learning to code impacts the acquisition of CT in young children, it may be possible to improve teaching methods designed to promote the development of these reasoning skills (Nouri et al., 2020). The advent of the CAL curriculum and validated unplugged CT assessments for elementary school children provides a new opportunity to explore the interaction of coding education and CT. We set out to answer the following research question: How does a coding intervention impact young children's CT skills as measured by an unplugged CT assessment?

## 2. Method

The present study has a quasi-experimental longitudinal design. The intervention is a version of the CAL curriculum called "CAL-KIBO" that uses the KIBO robot to teach children programming and literacy concepts. It examines CT skills in children between ages 5 and 9 (first and second grade) before and after they participate in the CAL-KIBO curriculum. Grade-matched students who engage in their usual classroom activities without learning to code provide a comparison group for identifying incidental and/or maturation-related changes in CT skills. The *TechCheck* unplugged assessment is administered before and after the intervention to evaluate changes in CT. In the following section, we describe the specifics of the methods we employ.

### 2.1. The intervention: the CAL-KIBO curriculum

The CAL-KIBO curriculum is implemented using the KIBO robotics platform, a screen-free programmable robot that is developmentally appropriate for young children. Young children often learn to code using simple sequencing and graphical or tangible coding interfaces (Bers, 2020; Guzdial & Morrison, 2016; Jenkins, 2002; Resnick & Silverman, 2005; Strawhacker et al., 2017; Sullivan et al., 2015). KIBO is programmed with tangible wooden blocks that a child sequences and then scans using a barcode scanner embedded in the robot. Each block represents an action that the robot performs. The combination of KIBO's blocks, sensors, modules, and art platforms gives children a unique opportunity to not only explore programming concepts but also to use their creativity to create personally meaningful projects (see Fig. 1).



Fig. 1. The KIBO robot, programming blocks, parameter stickers, modules/sensors, and attachable art platforms.



KIBO has been shown to engage young children as young as four years old in expressive and creative coding (Elkin et al., 2016; Sullivan et al., 2015, 2017). The CAL-KIBO curriculum incorporates lessons and exercises that teach algorithms, modularity, hardware/software, control structures, debugging, representation, and design process.

The CAL-KIBO curriculum teaches coding as a symbolic system of representation for expressive purposes and not only problem-solving. CAL-KIBO includes time spent working with coding, game-play as well as an emphasis on activities involving social interactions, creativity and movement. Individual and group activities in this curriculum include warm-up games to playfully introduce or reinforce concepts, design challenges to solidify skills, free explorations to allow students to tinker and expand their skills, expressive explorations to promote creativity, writing activities and technology circles to share and reflect on activities. The curriculum was first created and tested with second graders and was then modified for first graders. Feedback from teachers, administrators and students was taken into account when designing this curriculum. The CAL-KIBO curriculum is aligned with the Common Core English Language Arts (ELA)/Literacy Framework, as well as Virginia CS Standards of Learning and other nationally recognized CS frameworks (ISTE Standards for Students, 2017; K-12 Computer Science Framework Steering Committee, 2016; Massachusetts Department of Elementary and Secondary Education, 2016; National Governors Association Center for Best Practices & Council of Chief State School Officers, 2010; Virginia Department of International Society for Technology in Education, 2017).

The second grade CAL-KIBO curriculum consisted of 12 1-h lessons. Lessons were designed to be carried out in 1–2 h of instruction each week over 6–7 consecutive weeks. Each lesson consisted of structured KIBO challenges, opportunities for free exploration and writing activities. The advanced programming concepts in this curriculum included repeat loops, the use of light and distance sensors, and conditionals. The final lesson involved a multi-day project based on the popular children’s book *Where the Wild Things Are* by Maurice Sendak, which was referenced at several points throughout the curriculum. This book was chosen because it fosters discussion and creative thinking and allows teachers to integrate literacy and computer science concepts into their lessons.

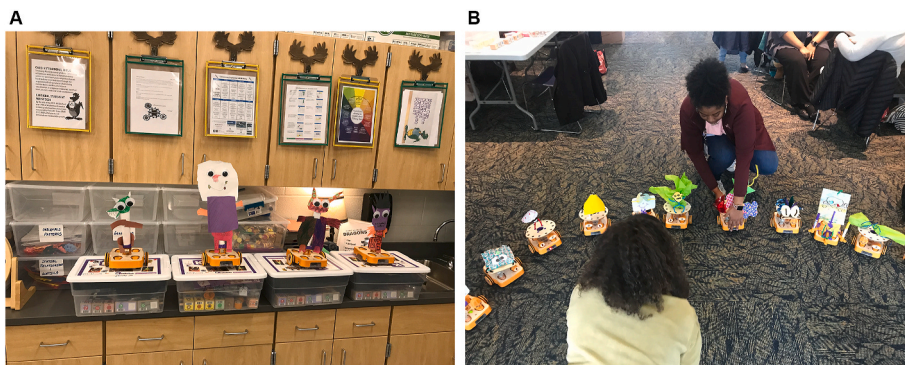
The first grade CAL-KIBO curriculum followed the same implementation timeline and used the same story *Where the Wild Things Are* and covered much of the same KIBO concepts but did not cover conditional statements. Additionally, 3 h of additional lesson time were added to the original 12-h curriculum based on teacher feedback. The “Wild Rumpus” compositional activity was omitted so that students could focus more on programming and student-centered discussions with KIBO. First grade teacher and classroom support materials were enhanced to better assist teachers in implementing the curriculum.

An example of a CAL lesson involves one of the main scenes in *Where the Wild Things Are* consisting of six pages of illustrations showing the main character Max participating in a “Wild Rumpus Party.” Students were asked to write a creative composition about what would happen at their own Wild Rumpus Party. The class then discussed their compositions as a group and collaborated with one another to decide whether or not what they had written about could be rendered as a program for KIBO to perform. Children then programmed the KIBO to perform their Wild Rumpus party activities (see Fig. 2A). For example, one child wrote that her KIBO would sing karaoke and dance. The child used stickers corresponding to those on the KIBO blocks to plan her program and subsequently programmed KIBO using actual programming blocks and a recording of her own singing made using the KIBO sound recorder module.

## 2.2. Participants

Participants in this study were first and second graders from an urban school district in Norfolk, Virginia. Students were from military and non-military families with a mixture of different racial/ethnic and socio-economic backgrounds (Table 3). Ten schools were invited to participate in this study. Eight of the ten schools received a grant from the U.S. Department of Defense and were chosen to receive the CAL-KIBO curriculum. Two additional schools were included for comparison purposes. Students from the No-CAL control schools followed a standard curriculum without exposure to CAL or coding. No-CAL students underwent assessments at comparable time intervals to the CAL schools. The No-CAL schools had similar overall demographics to the schools that received CAL (Table 3).

Among the eight schools invited to implement the CAL curriculum, two schools contributing first graders did not participate due to administrative and/or staffing issues. Among the No-CAL schools, one first grade class inadvertently received coding instruction during



**Fig. 2.** A: Students’ CAL-KIBO final projects. B: KIBOs decorated by teachers at the CAL-KIBO training, Note. Photograph 2A courtesy of Angela de Mik, Norfolk Public Schools.

**Table 3**  
Demographics of the study population.

	All CAL	All No-CAL	Grade 1 CAL	Grade 1 No-CAL	Grade 2 CAL	Grade 2 No-CAL
Number of students	667	181	271	71	396	110
Mean Age (Years)	7.41	7.38	6.23	6.28	7.56	7.61
Age Range (Years)	5–9	6–9	5–8	6–7	7–9	7–9
Gender						
Male (%)	47.20	42.54	48.34	43.66	46.46	41.81
Female (%)	51.87	56.35	50.92	56.34	52.53	56.36
Not specified (%)	0.93	1.01	0.74	0	1.01	1.82
Race						
Black/African American (%)	41.25	53.59	35.42	57.75	45.21	50.91
Hispanic (%)	10.19	14.26	10.33	16.90	10.10	12.73
Mixed (%)	8.54	5.52	9.59	5.63	7.83	5.45
White (%)	36.58	24.97	40.59	18.32	33.83	29.09
Asian/Pacific Islander (%)	2.99	1.66	3.32	1.40	2.78	1.82
Native American (%)	0.45	0	0.74	0	0.25	0

the study window in violation of the study protocol and was excluded from the analysis.

### 2.3. Inclusion criteria

Inclusion criteria for this study were: 1. parental opt-out consent and child assent; 2. adequate English language skills to participate in the curricular activities and study assessment. Inclusion criterion for the main analysis was the completion of baseline and end point *TechCheck*. Inclusion criterion for the baseline analysis was the completion of the pre-CAL *TechCheck* assessment.

### 2.4. Professional development

Educators attended a full day, CAL-KIBO training led by multiple researchers where they participated in hands-on play with the KIBO robot and were introduced to the CAL approach and curriculum. At the training, teachers participated in activities from the curriculum such as creating their own *Where the Wild Things Are* KIBO final projects (see Fig. 1B). Opportunities were provided to practice and plan for classroom implementation. Teachers were given hard copies of the curriculum and children's books to aid their instruction as well as online resources such as videos of others teaching the curriculum, links to the curriculum lessons, and lesson slides. Ongoing professional development and support was given to educators through phone calls with researchers and in-person assistance from administration staff, researchers, and instructional technology resource teachers.

Instructional Technology Resource Teachers (ITRTs) from the school district attended both the CAL-KIBO training and separate in-person assessment workshops. At the assessment workshops, ITRTs were taught to administer *TechCheck* including what to do in various scenarios (e.g., a child needing to leave the room, or asking them if they got the correct answer). ITRTs were given time to practice administration. A log was created to keep track of which classes received assessments and when. Additionally, throughout the study ITRTs and researchers engaged in multiple phone conferences to provide feedback on assessment administration.

### 2.5. Computational thinking assessment

The *TechCheck* assessment used in this study consists of fifteen multiple-choice questions. *TechCheck* is considered an “unplugged” assessment because its challenges probe CT but do not require the use of technology or knowledge of computer programming to be completed. In the present study, *TechCheck* was administered using computers and tablets rather than pencil and paper. However, it is the content rather than the mode of administration that leads to the characterization of *TechCheck* as an unplugged assessment. The child responds to prompts on *TechCheck* by clicking on one of four options. Each correct response is awarded one point, with a maximum total score of 15 points. Two practice questions are included in the beginning of the assessment to familiarize students with the format but are not included in the scoring. All questions must be answered to complete the assessment. The *TechCheck* assessment typically takes an average of 13 min for children to complete. *TechCheck* was previously validated with a sample ( $N = 768$ ) of 5-9-year-old children in first and second grade (Relkin et al., 2020). The assessment showed good discrimination of children between different skill levels and an adequate difficulty level for first grade. The difficulty level for second graders was low and a ceiling effect was evident for the highest performers. Children's scores on *TechCheck* correlated moderately and positively ( $r = 0.53$ ) with a CT measure (TACTIC-KIBO) that requires knowledge of coding with the KIBO robot (Relkin et al., 2020).

*TechCheck* probes six of the seven powerful ideas from computer science described by Bers (2018) as developmentally appropriate for children ages 4–9. This includes algorithms, modularity, control structures, representation, hardware/software, and debugging. Design process, the seventh powerful idea, was not included in *TechCheck* because it is an inherently open-ended process that cannot be readily measured in a multiple-choice format assessment (Relkin et al., 2020). A variety of different tasks are used to probe the six CT domains: sequencing challenges, shortest path puzzles, missing symbol series, object decomposition, obstacle mazes, symbol shape puzzles, identifying technological concepts, and symmetry problems (see Appendix).

## 2.6. Procedure

After attending in-person professional development, the teachers were given two weeks to prepare their classroom schedules. One week prior to initiating the curriculum, the *TechCheck* assessment was administered by one of eight ITRT proctors (one per school). The endpoint *TechCheck* was given after the full curriculum had been taught.

ITRTs were trained to administer the *TechCheck* assessments consistently. Entire classrooms were tested together on individual tablets. Before children arrived, ITRTs prepared enough devices for the classroom and opened the *TechCheck* assessment application saved to the desktop. ITRTs first established rapport with children, then asked children for their assent to participate. Since the *TechCheck* assessment is designed for use in children who may be pre-literate or marginally literate, administrators were instructed to project a copy of the assessment onto a board and read each question out loud to the students twice. There were two practice questions that the classroom did as a group to ensure that children knew how to use the application and select answers using the interface. Students were then told to work individually and were given up to 1 min to answer each question. Each question required a response and children were instructed to guess if they did not know the answer.

## 2.7. Data analysis

Statistical analyses were conducted in R (Version 3.6.1, R Core Team, 2019) using R Studio version 1.2 (R Core Team, 2019). To assess longitudinal changes over time, we analyzed the data with a General Linear Mixed Model (GLMM) using the “lme4” package (Bates et al., 2015) and the “lmerTest” package (Kuznetsova et al., 2017). Pre-analysis data screening showed adequate normality of the variables used in the models. The GLMM fixed effects were: CAL vs No-CAL status, age, self-reported gender, grade, and Baseline *TechCheck* score. The random effect was classroom/teacher. *P*-values were obtained by likelihood ratio tests for the full model with CAL and the model without CAL and by using the “summary” function on R studio. A post-hoc analysis of residuals showed adequate normality on histogram and P–P plots. VIF and Tolerance assumptions of multicollinearity were met. Although the majority of the data appeared to obey the assumptions of homoscedasticity, there was some sparseness of data at the lower end of the range of *TechCheck*

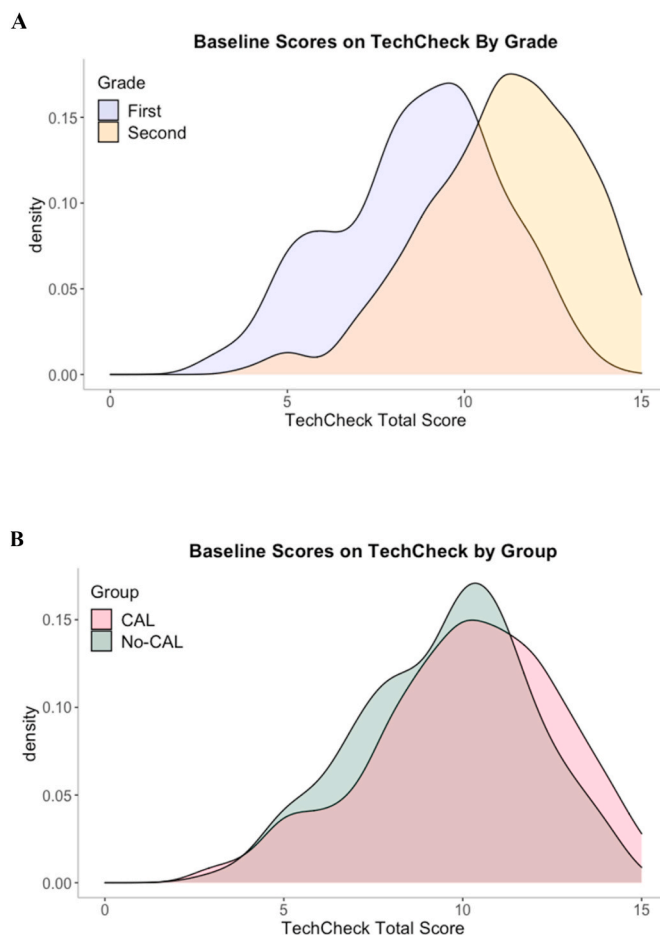


Fig. 3. The Distribution of Baseline *TechCheck* Scores by grade(3A) and by group (3B).

scores, so the assumption of homoscedasticity may not have been met. No influential outliers were identified.

We calculated the expected rate of change of *TechCheck* scores over time in the absence of a coding intervention by subtracting the mean baseline *TechCheck* score in first graders from the mean score in second graders and dividing that by the difference in mean ages between the two grades.

To further assess CAL-KIBO's contributions to the study outcomes, we conducted Bayesian Linear Mixed Modeling (BLMM) with the "BayesFactor" package (Morey & Rouder, 2018). BLMM was used in this study to supplement classical hypothesis testing since one cannot infer from this type of analysis whether the null hypothesis is true if, for example, results are non-significant (see Dienes, 2014). Bayesian analyses can provide information about the relative strength of the statistical evidence for both the null and alternative hypotheses. A Bayes factor is the likelihood ratio of one hypothesis divided by that of another hypothesis.

A post-hoc item analysis was carried out examining change in the percentage of correct responses in the six *TechCheck* CT domains. First, the percentages of correct responses on each of the 15 *TechCheck* items were calculated and then averaged within each CT domain. Baseline percentages were then subtracted from endpoint percentages to determine change over the course of the 7-week intervention for the students who received the CAL-KIBO curriculum (CAL) and those who did not (No-CAL). For comparison purposes, predicted change over seven weeks of typical development was calculated by subtracting the percentage of correct responses for all first graders at baseline from those of all second graders at baseline, and multiplying the resulting percentages by 7/67.8 to adjust for the 1.3-year average difference in age between the two grades.

### 3. Results

#### 3.1. Baseline score distributions

The distributions of *TechCheck* scores at baseline for first and second grades are shown in Fig. 3a. Scores were approximately normally distributed. A rightward skew is visible in the second grade distribution, consistent with the ceiling effect on *TechCheck* previously observed in second graders (Relkin et al., 2020). A Welch Two Sample *t*-test showed that there was a significant difference in baseline *TechCheck* scores between first ( $M = 8.45$  points,  $SD = 2.33$ ) and second ( $M = 10.99$  points,  $SD = 2.20$ ) grades;  $t(703.81) = 15.92$ ,  $p < .001$ . We used the mean difference in baseline scores (2.54 points) divided by the mean difference in the ages of first and second graders (1.30 years) to calculate the approximate expected change in *TechCheck* scores between first and second grade (1.95 points/year). Since most of the study participants (>75%) indicated they had little or no past coding experience, this rate of change can be taken to approximate maturation-related changes in CT skills over time.

Fig. 3b shows the distribution of scores for students in the CAL and No-CAL groups. A Welch Two Sample *t*-test showed that baseline performance was higher in the schools that received CAL ( $M = 10.09$  points,  $SD = 2.61$ ) compared to the No-CAL schools, ( $M = 9.50$  points,  $SD = 2.38$ ;  $t(307.73) = 2.93$ ,  $p < .01$ ). This difference in baseline scores occurred by chance rather than by design.

#### 3.2. Primary outcome

Students who received the CAL-KIBO curriculum (CAL group) improved on *TechCheck* ( $M_{change} = 0.94$ ,  $SD = 2.28$ ). Paired sample *t*-tests showed that the CAL group's change from baseline ( $M = 10.09$ ,  $SD = 2.61$ ) to endpoint ( $M = 11.03$ ,  $SD = 2.61$ ) was significant;  $t(666) = 10.55$ ,  $p < .001$ . Students in the No-CAL control group who engaged in typical classroom studies without coding instruction did

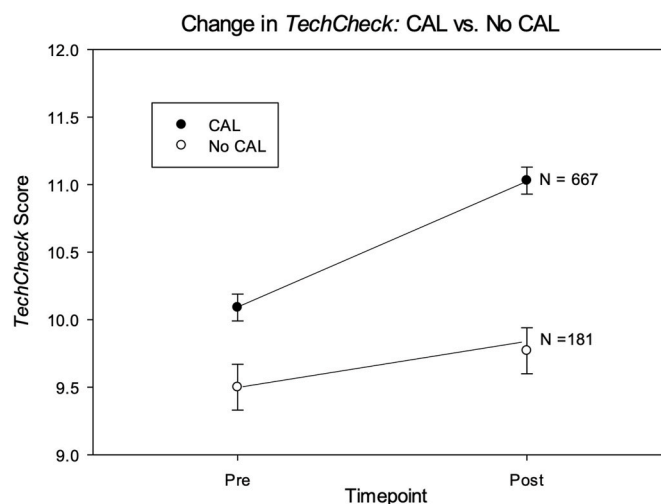


Fig. 4. Unadjusted mean change in *TechCheck* scores (+/- S.E.) from baseline (pre) to endpoint (post) in students receiving the CAL-KIBO coding curriculum ● versus control (No-CAL) students ○.

not significantly improve ( $M_{change} = 0.27$  points,  $SD = 2.04$ ) from baseline ( $M = 9.50$ ,  $SD = 2.38$ ) and the study endpoint ( $M = 9.77$ ,  $SD = 2.55$ ) assessments;  $t(180) = 1.81$ ,  $p = .07$ ) (see Fig. 4).

The observed change of 0.94 points in the CAL group equates to the increase in *TechCheck* scores that would be expected to occur over approximately 6 months without instruction based on the observed increase of 1.95 points per year in baseline scores. The change of 0.27 points in the No-CAL control group is consistent with the expected change over seven weeks, which is the actual interval over which the testing was performed.

Results were stratified by grade and paired sample t-tests were conducted. In first grade the CAL group significantly improved ( $M_{change} = 1.32$  points,  $SD = 2.31$ ) from baseline ( $M = 8.63$ ,  $SD = 2.35$ ) to endpoint ( $M = 9.95$   $SD = 2.34$ ) of the study;  $t(270) = 9.21$ ,  $p < .001$ . Likewise, the second grade CAL group significantly improved ( $M_{change} = 0.68$  points,  $SD = 2.23$ ) from baseline ( $M = 11.15$   $SD = 2.20$ ) to the study endpoint ( $M = 11.83$   $SD = 2.46$ );  $t(395) = 6.11$ ,  $p < .001$ . No significant improvements were found for the No-CAL control group in first grade ( $M_{change} = 0.01$ ,  $SD = 2.10$ ) from baseline ( $M = 8.03$ ,  $SD = 2.05$ ) to endpoint ( $M = 8.10$ ,  $SD = 2.27$ ) assessments;  $t(358.31) = 1.07$ ,  $p = .95$ ). A borderline significant improvement was found in the second grade No-CAL control group ( $M_{change} = 0.44$ ,  $SD = 2.00$ ) from baseline ( $M = 10.43$   $SD = 2.10$ ) to endpoint ( $M = 10.87$ ,  $SD = 2.07$ );  $t(109) = 2.34$ ,  $p = .05$ ), (see Table 4).

### 3.3. GLMM results

To take into account baseline differences and to evaluate the contribution of other effects such as age, gender, ethnicity and classroom differences, we used Generalized Linear Mixed Model (GLMM) analysis. For the two grades combined, the effects found to be significant on the GLMM included the intercept ( $p < .001$ ), CAL ( $p < .01$ ), grade ( $p < .01$ ) and baseline *TechCheck* score ( $p < .001$ ) (See Table 5 and Fig. 5). Gender and age did not significantly contribute to the model. This indicates that exposure to the CAL-KIBO curriculum was a significant predictor of endpoint *TechCheck* outcome, even when taking into account differences in baseline *TechCheck* performance and other effects. The Bayes factor for this model compared to a model without CAL was  $>100$ . This is classified as “decisive evidence” against the null hypothesis (Wetzels et al., 2011) and strongly supports CAL being a significant factor in the overall *TechCheck* outcomes. Interaction terms did not improve the model’s fit to the data and were therefore not included.

When data from the first grade students only were modeled, CAL ( $p < .01$ ) and baseline score ( $p < .001$ ) were significant effects. Gender, age, and the intercept were not significant. The Bayes factor for the model including CAL compared to one without CAL was  $>100$ , a level considered decisive evidence that exposure to CAL is a predictor of post-test CT scores. When data from second grade students were examined, gender was significant at the  $p < .05$  level, as were the baseline score ( $p < .001$ ) and the intercept ( $p < .001$ ). Notably, CAL did not reach significance in this model. The Bayes factor, in this case was 1.88, which is considered “anecdotal evidence” (Wetzels et al., 2011). Thus, in contrast to the results in first graders, we cannot say with certainty that exposure to CAL is a predictor of *TechCheck* score in second graders.

### 3.4. TechCheck item analysis

We sought to determine if changes in *TechCheck* scores after exposure to the CAL-KIBO curriculum were related to improved performance in specific CT domains. To do so, we carried out a post-hoc analysis of change in percentage of correct responses averaged over the questions within each CT domain. Fig. 6 shows the change in percentage of students whose scores improved from the study baseline to endpoint on each of the six domains measured by *TechCheck*.

All *TechCheck* CT domains showed change with typical development as calculated from differences between first and second grades at baseline. Hardware/software and debugging showed slightly less change with typical development than the other domains. There was likely a ceiling effect for these two domains since an average of 90% of students responded to hardware/software probes correctly and 87% responded correctly to debugging probes at baseline.

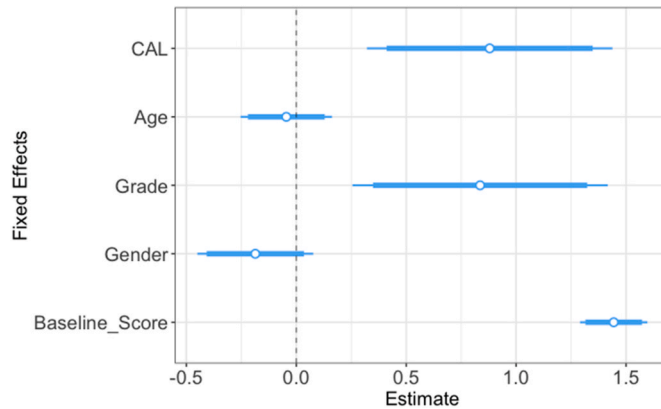
Across all six *TechCheck* domains, the average percentage increase in students responding correctly after the coding intervention was 6% for the CAL group and 2% for No-CAL controls. The largest percentage of student improvement in the CAL group was associated with the CT domains of Modularity, Algorithms and Representation, respectively (see Fig. 6).

**Table 4**  
*TechCheck* results for CAL and No-CAL Control Groups.

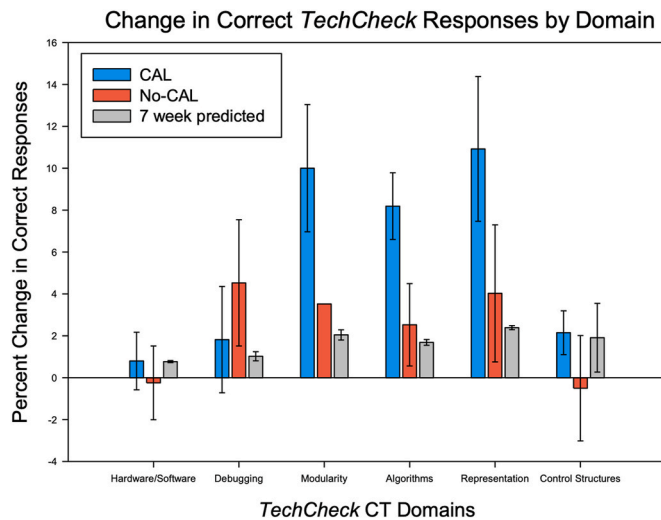
Group	N	<i>TechCheck</i> Baseline (M ± SD)	<i>TechCheck</i> End Point (M ± SD)	Mean Points Changed	Median Points Changed	Paired t-test p Value	Bayes Factor (Bayesian t-test)
All CAL	667	10.09± 2.61	11.03± 2.61	0.94	1	$p < .001$	$>1000$
All No-CAL	181	9.50±2.38	9.77±2.55	0.27	0	$p = .07$	0.42
CAL	271	8.63±2.35	9.95±2.34	1.32	1	$p < .001$	$>1000$
Grade 1							
No-CAL	71	8.06± 2.05	8.07± 2.28	0.01	0	$p = 0.95$	0.13
Grade 1							
CAL	396	11.15±2.20	11.83±2.46	0.68	1	$p < 0.001$	$<1000$
Grade 2							
No CAL Grade 2	110	10.43±2.10	10.87±2.07	0.44	.5	$p < 0.05$	1.42

**Table 5**  
GLMM results modelling exposure to CAL and outcome of *TechCheck*.

	Estimate	CI (95%)	Standard Error	T-value	DF	P value
Intercept	4.50	5.92	0.87	5.20	774.08	$p < .001$
CAL	0.87	1.33	0.29	3.05	56.13	$p < .01$
Age	-0.06	0.14	0.12	-0.46	817.23	$p = .05$
Grade	0.84	1.32	0.30	2.86	142.82	$p < .01$
Gender	-0.19	0.03	0.13	-1.40	808.23	$p = .36$
Baseline score	.57	0.61	.03	18.51	831.59	$p < .001$



**Fig. 5.** Magnitude of effect and 95% C.I for fixed effects in GLMM



**Fig. 6.** Changes in percent of students making correct responses in the six CT domains measured by *TechCheck*. For the CAL and No-CAL groups, colored bars represent the percent difference in correct *TechCheck* responses between the baseline and end of study assessments. The “7 week predicted” results are calculated from the percent difference in correct responses between first and second graders at their respective baseline assessments, multiplied by 7/67.8. Error bars indicate standard errors of the means.

#### 4. Discussion

This study provides empirical evidence, from a large-scale, quasi-experimental study, that teaching first and second grade students (ages 5–9) to code through the CAL-KIBO curriculum can accelerate the acquisition of CT skills. The current study is the first to use *TechCheck* to document longitudinal changes in CT in association with an educational intervention. The CAL-KIBO curriculum teaches coding without explicitly providing children with unplugged CT challenges of the kind encountered in *TechCheck*. Consequently, the observed increase in *TechCheck* scores following the CAL-KIBO coding curriculum can be taken to reflect improvements in CT skills rather than practice effects or other assessment artifacts. This conclusion is further bolstered by the observation that the No-CAL group



did not demonstrate comparable improvement on *TechCheck* over the same time interval.

Most past studies in young children that explore the effects of learning to code on CT skills used coding exercises, interviews or measures such as neuropsychological tests as the means of assessment (Grover et al., 2014; Yadav et al., 2017). By utilizing an easily administered unplugged CT assessment, we were able to conduct the first longitudinal large-scale study in early elementary school children with a sizable non-coding control group. Our approach allowed assessment of baseline CT skills in these children regardless of their past coding experience and avoided conflating coding abilities with CT skills. The study also demonstrates the potential utility of *TechCheck* for assessing young children's CT in routine education settings.

The mean change in *TechCheck* scores in students exposed to the CAL-KIBO coding curriculum was slightly less than one point out of a maximum score of 15 points. This magnitude of improvement after seven weeks of coding instruction is consistent with the estimated change in baseline *TechCheck* scores in the absence of coding instruction over approximately six months. This change is comparable in magnitude to the improvements in executive functions reported by Arfé et al. (2019) in a group of  $n = 17$  second graders, in whom exposure to the [code.org](#) curriculum for one month resulted in improvements in executive functions equal to 7 months in age-matched students who received non-coding STEM instruction (Arfé et al., 2019).

Our data also indicate that young children's performance on a CT assessment can improve in the course of typical development. CT may improve as a consequence of brain maturation, practice effects, as a result of learning in other disciplines and through various life experiences. The present study does not allow us to say whether CT acquired in the context of learning to code is the same or different in nature from that gained through typical development or non-coding experiences. However, by accelerating the acquisition of CT, coding interventions in early childhood may exert long-term benefits analogous to the improved academic outcomes associated with early acquisition of literacy skills (Heckman & Masterov, 2007; Stanovich, 1986, 2000).

In this study, first graders in the CAL group improved more on *TechCheck* than the CAL group second graders. It is possible that the higher baseline *TechCheck* scores in second graders reduced the range of possible improvement compared to first graders. Previous psychometric analysis of the *TechCheck* assessment revealed a less-than-optimal difficulty level for higher performing second graders (Relkin et al., 2020) which may have led to a ceiling effect in the present study. Another possible explanation for the observed differences between grades would be if the CAL-KIBO curriculum was more effective in first graders. This possibility can neither be confirmed nor ruled out based on the currently available data.

There were significant differences at baseline in the *TechCheck* scores of the CAL-KIBO versus the No-CAL control groups. We carried out GLMM modeling to take those differences into account and evaluate the effects of other demographic and environmental variables on the study's outcomes. The results of the GLMM analyses indicate that exposure to CAL-KIBO was a highly significant predictor of *TechCheck* outcome even when taking the other variables into account. GLMM analysis did not confirm a significant effect for CAL in second graders which we believe is due at least in part to a ceiling effect in that grade discussed above.

The inclusion of a control group that did not receive coding instruction is important for several reasons. This is the first study in which *TechCheck* was administered serially to large numbers of students, and it was therefore important to control for the possibility of a learning effect from repeated exposure to the assessment. The results from the No-CAL control group suggest that any learning effects from repeated testing did not profoundly affect the study's outcome. The inclusion of a No-CAL control group also allowed for observation of changes in *TechCheck* performance related to the maturation of students over the time interval of the study.

Our findings suggest that some students who received coding instruction were able to transfer the knowledge they gained from coding into CT skills useful for solving unplugged problems. Although *TechCheck* is not designed to quantitatively assess CT skills in specific CT subdomains, post hoc analysis suggests that the domains of CT that improved most after the CAL-KIBO curriculum were algorithms, modularity, and representation. These are similar in nature to the domains identified in surveys of educators as being enhanced in young children who learn to code (Nouri et al., 2020).

It is worthwhile considering why the items designed to probe algorithms, modularity and representation may have shown the highest percentage of improvers. CAL-KIBO emphasizes the relationship between CS concepts and literacy, drawing on children's stories as inspiration for programming projects and other exercises. As students engage in KIBO coding activities, they learn programming fundamentals such as recognizing the relationship between symbols on the KIBO programming blocks and concrete actions (e.g., movement of the robot). They also learn that using certain blocks results in tangible actions while others exert effects that are not as readily visible (e.g., conditionals). As children learn these coding fundamentals, the CAL-KIBO curriculum challenges them to achieve goals such as programming simulations of storybook characters and robotic re-enactments of story elements from children's literature. Participating in these challenges requires learning about symbolism (representation), decomposition of multistep processes into executable steps (modularity) and sequencing multiple steps to achieve a desired set of actions (algorithms). By recognizing patterns that repeat within and across these creative activities, children may acquire CT skills in the above-mentioned domains. Hands-on experience with creative activities may better enable students to generalize the knowledge they acquire from coding and apply it to new situations (Basawapatna et al., 2010). Another possible reason for seeing greater effects in algorithms, modularity and representation could be if *TechCheck* is inherently more sensitive to changes in these domains. Since experience with *TechCheck* in longitudinal studies is currently limited, additional studies will be needed to examine this possibility.

*TechCheck's* probes of hardware/software and control structures did not show differences between the CAL and No-CAL groups. There was likely a ceiling effect for these domains since close to 90% of students across the two grades responded correctly at baseline. This left little room for improvement after the coding intervention. The *TechCheck* probes for control structures involve navigating through a maze with obstacles by following a set of conditional instructions. Young children tend to have trouble learning conditionals (Elkin et al., 2014; Strawhacker & Bers, 2015) and it is possible that this concept was not fully developed in this version of the CAL-KIBO curriculum.

The change score for debugging showed a numerically greater change for the No-CAL group than the CAL group. However, the

difference was within the variance of the results. The debugging probes in this version of *TechCheck* involve correcting an unbalanced seesaw. It is possible that the children in the No-CAL group learned problem-solving through another school subject that increased their debugging skills. In interpreting all of these post-hoc, domain-specific results, it should be kept in mind that *TechCheck* is designed as a composite measure of CT across all six domains, not as a way to precisely quantify performance in individual CT domains.

At least one author has argued that transfer of CT skills to other disciplines may not occur effectively if children are introduced to problem-solving exclusively in the context of learning to code (Curzon, 2013). Curzon (2013) suggested that teaching young children programming primarily helps them develop coding-related reasoning rather than thinking skills in other domains. By introducing unplugged activities, Curzon argued that one can invoke more powerful skills of improved logical thinking and problem solving. However, participation in unplugged activities alone may not foster the development of higher reasoning skills. Thies and Vahrenhold (2012, 2013) studied the impact of *CSunplugged* in elementary and middle school children using qualitative and semi-quantitative assessments. They did not find evidence of improvements in higher-level reasoning skills and suggested that exposure to unplugged activities alone does not necessarily lead to a generalization of learning or promote the development of higher reasoning skills. While other studies have concluded that unplugged activities alone are ineffective (Black et al., 2013), some have found unplugged activities to be equally effective to coding in promoting CT (Hermans & Aivaloglou, 2017; Metin, 2020; Wohl et al., 2015). Some authors have argued that CS education is more effective when lessons include actual technology and coding in addition to unplugged activities (Bers, 2020; Huang & Looi, 2020; Thies & Vahrenhold, 2012, 2013). More research is needed to establish whether this is true and if so, to clarify the optimal approach to integrating these elements into a CS curriculum.

The importance of knowledge transfer and generalization of learning in the emergence of CT from coding education has been emphasized by several investigators (Angeli et al., 2016; Grover et al., 2015; Ioannidou et al., 2011; Reppenning et al., 2015). “Near transfer” of knowledge refers to circumstances in which there is a generalization of learning sufficient to facilitate learning of new material that is similar in nature to the original learning materials. “Far transfer” refers to the adaptation of knowledge from a learned skill to help solve entirely new types of problems which may be in completely different disciplines (Reschly & Robinson-Zaňartu, 2000). While the unplugged challenges in *TechCheck* were not actual coding exercises, they were selected as probes from the same domains of CT as are embodied in the CAL-KIBO coding curriculum. In this context, the improvement in unplugged problem-solving skills observed in the CAL group can be considered a form of near transfer of knowledge.

#### 4.1. Limitations

Our initial intention was to carry out this study with kindergarten students as well as first and second graders. Although kindergarten teachers attended professional development, we were unable to implement the kindergarten CAL-KIBO curriculum due to school closure from the COVID-19 pandemic.

The No-CAL and CAL groups were not chosen at random. CAL schools were invited to participate first and No-CAL schools were added later based on having similar school level demographics to the CAL group. We did not include specific measures of SES in this study. Socioeconomic status (SES) can impact the acquisition of coding skills and CT (Google & Gallup, 2016; Scherer & Siddiq, 2019). We cannot rule out the possibility that between-group differences in SES contributed to the observed *TechCheck* outcomes.

*TechCheck* is a relatively new screening instrument and the version used in this study did show ceiling effects that may have reduced the magnitude of the observed outcomes, particularly in second graders. We are in the process of validating a revised version of the *TechCheck* assessment designed to more effectively discriminate a range of CT skill levels across three grades (K, 1, 2). We recognize that children’s engagement in open-ended creativity and self-expression are integral to the learning and development of CT. However, *TechCheck* does not assess students in this domain owing to limitations imposed by its multiple-choice format. In the future, other more open-ended forms of assessment may be beneficial to implement in combination with *TechCheck* to get a more comprehensive picture of the child’s CT development.

Although *TechCheck* successfully detected improvement of CT skills in this study, the percentage of students showing improvement was relatively small. We hope that the revisions to the assessment that have been made subsequent to this study will render it even more sensitive to change particularly in the domains that showed ceiling effects. In addition, we hope future enhancements to the coding curriculum will lead to greater improvements in CT skills after coding instruction. Due to timing restrictions with the second-grade public schools that participated, we were not able to use the full version of the CAL-KIBO curriculum that we had originally intended to implement. Our original version for second graders had an additional 24 lessons (24 h) of instruction. Future iterations of the CAL-KIBO curriculum will be extended and should allocate more time to help scaffold abstract and advanced programming concepts.

#### 4.2. Future directions

This study has implications for the design of future coding and CT curricula for young children. Currently, many CS professional development programs for teachers focus on the syntax and implementation of programming languages rather than techniques that foster CT skill acquisition and authentic learning in other content areas (Sands et al., 2018). It may be beneficial for professional development to emphasize teaching methods that foster students’ self-expression and creativity through coding to better promote young children’s CT skills. Future studies comparing technically focused coding programs to curricula like CAL-KIBO that integrate literacy and creativity components can help to establish best practices for young children.

We believe it is important that these findings be extended to other cohorts and grades, as well as other curricula and educational contexts. The *TechCheck* assessment has been translated into multiple languages (i.e., Spanish, Chinese, Turkish) and is currently being



administered in diverse research and educational settings. Future studies should explore its use in children from various cultures and neuro-diverse children.

More work is needed to understand best practices for teaching CT to young children. Future longitudinal studies should compare different the effects of different coding curricula. A logical next iteration might be to compare the effects of CAL-KIBO with those of a conventional coding curriculum and one that combines coding with unplugged activities. By this approach, we can hope to learn whether young children best acquire CT when they are taught using platform-specific coding exercises, unplugged activities or a hybrid approach using both methods.

### Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marina Umaschi Bers co-founded the company Kinderlab that manufactures the KIBO robot and has entered into a licensing agreement with Tufts University for the technology. The research presented is not expected to have any direct financial impact. No other authors have competing interest relating to this work.

### Acknowledgements

The authors would like to thank the project coordinator Angela de Mik as well as members of the DevTech research group that contributed to this project (Madhu Govind, Ziva Hassenfeld, Pat Nero, Maya Morris, Ari Lerner, and Jaclyn Tsiang) who this work would not be possible without. The authors would also like to thank the Instructional Technology Resource Technicians. Lastly, we would like to thank all of the teachers, administrators, and children involved in this study.

### Appendix A. Supplementary data

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.compedu.2021.104222>.

### Funding

This work was supported by the Department of Defense Education Activity (DoDEA) “Operation: Break the Code for College and Career Readiness”. Unique Entity Identifier: “WORLDCL10”.

### Credit author statement

Emily Relkin: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration  
 Laura de Ruiter: Conceptualization, Validation, Formal analysis, Data curation, Writing- Original Draft, Writing – review & editing, Visualization, Supervision  
 Marina Bers: Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration.

### References

- Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, 55(7), 832–835. <https://doi.org/10.1093/comjnl/bsx074>
- 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. Retrieved from <https://www.jstor.org/stable/10.2307/jeductechsoci.19.3.47>.
- Arfè, B., Vardanega, T., Montuori, C., & Lavanga, M. (2019). Coding in primary grades boosts children’s executive functions. *Frontiers in Psychology*, 10(2713). <https://doi.org/10.3389/fpsyg.2019.02713>
- Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning and Leading with Technology*, 38(6), 20–23. Retrieved from <https://id.iste.org/docs/learning-and-leading-docs/march-2011-computational-thinking-11386.pdf>.
- Barrouillet, P., & Lecas, J. (1999). Mental models in conditional reasoning and working memory. *Thinking & Reasoning*, 5(4), 289–302.
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Inroads*, 2(1), 48–54. <https://doi.org/10.1145/1929887.1929905>
- Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 224–228). <https://doi.org/10.1145/1822090.1822154>. ACM.
- Basu, S., Biswas, G., Sengupta, P., Dicks, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students’ challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*, 11(1), 13. <https://doi.org/10.1186/s41039-016-0036-2>
- Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1), 1–48. <https://doi.org/10.18637/jss.v067.i01>
- Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work? In H.-J. In H.-J. Böckenbauer, D. Komm, & W. Unger (Eds.), *Adventures between lower bounds and higher altitudes: Essays dedicated to Juraj Hromkovič on the occasion of his 60th birthday* (pp. 497–521). [https://doi.org/10.1007/978-3-319-98355-4\\_29](https://doi.org/10.1007/978-3-319-98355-4_29)
- Bers, M. U. (2018). Coding as a playground: Programming and computational thinking in the early childhood classroom. *Routledge*.
- Bers, M. U. (2019). Coding as Another Language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, 6(4), 499–528. <https://doi.org/10.1007/s40692-019-00147-3>
- Bers, M. U. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom* (2nd ed.). New York, NY: Routledge Press.

- Black, J., Brodie, J., Curzon, P., Mykietiaki, C., McOwan, P. W., & Meagher, L. R. (2013). Making computing interesting to school students: Teachers' perspectives. In *Proceedings of the 18th ACM conference on innovation and technology in computer science education* (pp. 255–260). Association for Computing Machinery.
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada, 1 p. 25*. Retrieved from [https://web.media.mit.edu/~kbrennan/files/Brennan\\_Resnick\\_AERA2012\\_CT.pdf](https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf).
- Bruner, J. S., Olver, R., & Greenfield, P. (1966). *Studies in cognitive growth*. New York: Wiley.
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers in Education*, 109, 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>
- Clements, D. H., Battista, M. T., & Sarama, J. (2001). Logo and Geometry. In E. Yackel (Ed.), *Journal for research in mathematics education monograph series*, 10. <https://doi.org/10.2307/749924>
- Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, 76(6), 1051–1058. <https://doi.org/10.1037/0022-0663.76.6.1051>
- CSTA. (2011). *Operational definition of computational thinking for K–12 education*, 2011. Retrieved from <http://www.csta.acm.org/Curriculum/sub/CompThinking.html>.
- Cuny, J., Snyder, L., & Wing, J. M. (2010). *Demystifying computational thinking for non-computer scientists*. Unpublished manuscript in progress, referenced in <http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>.
- Curzon, P. (2013). cs4fn and computational thinking unplugged. In *Proceedings of the 8th workshop in primary and secondary computing education* (pp. 47–50). ACM. <https://doi.org/10.1145/2532748.2611263>.
- Dienes, Z. (2014). Using Bayes to get the most out of non-significant results. *Frontiers in Psychology*, 5(781). <https://doi.org/10.3389/fpsyg.2014.00781>
- Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice*, 13, 153–169.
- Elkin, M., Sullivan, A., & Bers, M. U. (2016). Programming with the KIBO robotics kit in preschool classrooms. *Computers in the Schools*, 33(3), 169–186. <https://doi.org/10.1080/07380569.2016.1216251>
- Feldman, D. H. (2004). Piaget's stages: The unfinished symphony of cognitive development. *New Ideas in Psychology*, 22, 175–231. <https://doi.org/10.1016/j.newideapsych.2004.11.005>
- Fields, D. A., Lui, D., & Kafai, Y. B. (2019). Teaching computational thinking with electronic textiles: Modeling iterative practices and supporting personal projects in exploring computer science. In *Computational thinking education* (pp. 279–294). Singapore: Springer. [https://doi.org/10.1007/978-981-13-6528-7\\_16](https://doi.org/10.1007/978-981-13-6528-7_16).
- Flavell, J. H., Miller, P. H., & Miller, S. A. (1993). *Cognitive development* (3rd ed.). Prentice Hall. NJ.
- Fraillon, J., Ainley, J., Schulz, W., Duckworth, D., & Friedman, T. (2018). *International computer and information literacy study: ICILS 2018: Technical report*.
- Goldstein, J., & Flake, J. K. (2016). Towards a framework for the validation of early childhood assessment systems. *Educational Assessment, Evaluation and Accountability*, 28(3), 273–293. <https://doi.org/10.1007/s11092-015-9231-8>
- Google for Education. (2010). *Exploring computational thinking*. Retrieved from [www.google.com/edu/resources/programs/exploring-computational-thinking/index.html#lhome](http://www.google.com/edu/resources/programs/exploring-computational-thinking/index.html#lhome).
- Google & Gallup. (2016). *Diversity gaps in computer science: Exploring the underrepresentation of girls, blacks and hispanics*. Retrieved from <https://services.google.com/fh/files/misc/diversity-gaps-in-computer-science-report.pdf>.
- Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 57–62). ACM. <https://doi.org/10.1145/2591708.2591713>.
- 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>
- Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199–237. <https://doi.org/10.1080/08993408.2015.1033142>
- Guzdial, M., & Morrison, B. (2016). Seeking to making computing education as available as mathematics or science education. *Communications of the ACM*, 59(11), 31–33. <https://doi.org/10.1145/3000612>
- Hassenfeld, Z. R., Govind, M., de Ruiter, L. E., & Bers, M. U. (2020). If you can program, you can write: Learning introductory programming across literacy levels. *Journal of Information Technology Education: Research*, 19, 65–85. <https://doi.org/10.28945/4509>
- Heckman, J., & Masterov, D. (2007). The productivity argument for investing in young children. *Review of Agricultural Economics*, 29(3), 446–493.
- Hermans, F., & Aivaloglou, E. (2017). To scratch or not to scratch?: A controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 49–56). Association for Computing Machinery.
- Huang, W., & Looi, C.-K. (2020). A critical review of literature on “unplugged” pedagogies in K-12 computer science and computational thinking education. *Computer Science Education. Advance online publication*. <https://doi.org/10.1080/08993408.2020.1789411>
- International Society for Technology in Education. (2017). *ISTE standards for students*. Retrieved from <https://www.iste.org/standards/for-students>.
- Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H., & Basawapatna, A. (2011). *Computational thinking patterns. Paper presented at the annual meeting of the American educational research association*. New Orleans, LA. Retrieved from <https://files.eric.ed.gov/fulltext/ED520742.pdf>.
- Janveau-Brennan, G., & Markovits, H. (1999). The development of reasoning with causal conditionals. *Developmental Psychology*, 35(4), 904–911.
- Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd annual. Conference of the LTSN centre for information and computer sciences* (pp. 53–58). Leeds, UK. Retrieved from <http://www.psy.gla.ac.uk/~steve/loaled/jenkins.html>.
- K-12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. Retrieved from <https://k12cs.org>.
- Kazakoff, E. R., & Bers, M. U. (2014). Put your robot in, Put your robot out: Sequencing through programming robots in early childhood. *Journal of Educational Computing Research*, 50(4), 553–573. <https://doi.org/10.2190/EC.50.4.f>
- Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). lmerTest package: Tests in linear mixed effects models. *Journal of Statistical Software*, 82(13), 1–26. <https://doi.org/10.18637/jss.v082.i13>
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32–37. <https://doi.org/10.1145/1929887.1929902>
- Lockwood, J., & Mooney, A. (2018). Computational thinking in education: Where does it fit? A systematic literary review. *International Journal of Computer Sciences and Engineering Systems*, 2(1), 41–60. <https://doi.org/10.21585/ijcses.v2i1.26>
- 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>
- Massachusetts Department of Elementary and Secondary Education. (2016). *Massachusetts digital literacy and computer science (DLCS) curriculum framework June 2016*. <http://www.doe.mass.edu/frameworks/dlcs.pdf>.
- McDevitt, T. M., & Ormrod, J. E. (2002). *Child development and education*. Upper Saddle River, NJ: Merrill/Prentice Hall.
- McMillan, J. H. (2013). *Classroom assessment: Principles and practice for effective instruction* (6th ed.). Boston: Pearson/Allyn and Bacon.
- Metin, S. (2020). Activity-based unplugged coding during the preschool period. *International Journal of Technology and Design Education*, 1–17.
- Mioduser, D., Levy, S. T., & Talis, V. (2009). Episodes to scripts to rules: Concrete-abstractions in kindergarten children's explanations of a robot's behavior. *International Journal of Technology and Design Education*, 19(1), 15–36.
- Morey, R., & Rouder, J. (2018). *BayesFactor: Computation of Bayes factors for Common designs*. R package version 0.9.12-4.2 <https://CRAN.R-project.org/package=BayesFactor>.
- Moyer, K., & Gilmer, B. V. H. (1953). The concept of attention spans in children. *Elementary School Journal*, 54(1), 464. <https://doi.org/10.1086/458623>
- Muller, U., Overton, W. F., & Reese, K. (2001). Development of conditional reasoning: A longitudinal study. *Journal of Cognition and Development*, 2(1), 27–49.

- Mullis, I. V., & Martin, M. O. (2019). *PIRLS 2021 assessment frameworks. International association for the evaluation of educational achievement. Herengracht 487*. Amsterdam, 1017 BT, The Netherlands. Retrieved from <https://eric.ed.gov/?id=ED606056>.
- National Governors Association Center for Best Practices & Council of Chief State School Officers. (2010). *Common Core state standards*. Washington, DC. Retrieved from <http://www.corestandards.org/about-the-standards/branding-guidelines/>.
- National Research Council. (2011). *Report of a workshop on the pedagogical aspects of computational thinking*. Washington, DC: National Academies Press.
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, 11(1), 1–17. <https://doi.org/10.1080/20004508.2019.1627844>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Piaget, J. (1953). *The origins of intelligence in the child*. London: Routledge and Kegan Paul.
- Piaget, J. (1971). Developmental stages and developmental processes. In D. R. Green, M. P. Ford, & G. B. Flamer (Eds.), *Measurement and Piaget* (pp. 172–188). New York: McGraw-Hill.
- Portelance, D. J., Strawhacker, A., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*. <https://doi.org/10.1007/s10798-015-9325-0>
- R Core Team. (2019). *R: A language and environment for statistical computing. R foundation for statistical computing*. Vienna: Austria. <https://www.R-project.org/>.
- Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*. <https://doi.org/10.1007/s10956-020-09831-x>
- Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Her Many Horses, I., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K., & Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education*, 15(2). <https://doi.org/10.1145/2700517>
- Reschly, D. J., & Robinson-Zanartu, C. (2000). Evaluation of aptitudes. *Handbook of Psychological Assessment*, 183–202.
- Resnick, M., & Siegel, D. (2015). A different approach to coding. *International Journal of People-Oriented Programming*, 4(1), 1–4. <https://doi.org/10.4018/IJPOP.2015010101>
- Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceeding of the 2005 conference on interaction design and children - IDC '05* (pp. 117–122). <https://doi.org/10.1145/1109540.1109556>
- Roman-Gonzalez, M., Moreno-Leon, J., & Robles, G. (2019). Combining assessment tools for a comprehensive evaluation of computational thinking interventions. In *Computational thinking education* (pp. 79–98). Singapore: Springer. Retrieved from [https://link.springer.com/chapter/10.1007/978-981-13-6528-7\\_6](https://link.springer.com/chapter/10.1007/978-981-13-6528-7_6).
- Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, 72, 678–691. <https://doi.org/10.1016/j.chb.2016.08.047>
- Roman-Gonzalez, M., Perez-Gonzalez, J. C., Moreno-Leon, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the computational thinking test. *International Journal of Child-Computer Interaction*, 18, 47–58. <https://doi.org/10.1016/j.ijcci.2018.06.004>
- Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking: Foundations and research highlights. In *Computational thinking in the STEM disciplines: Foundations and research highlights* (pp. 151–164). [https://doi.org/10.1007/978-3-319-93566-9\\_8](https://doi.org/10.1007/978-3-319-93566-9_8)
- Sattler, J. M. (2014). In J. M. Sattler (Ed.), *Foundations of behavioral, social and clinical assessment of children*. Publisher, Incorporated.
- Scherer, R., & Siddiq, F. (2019). The relation between students' socioeconomic status and ICT literacy: Findings from a meta-analysis. *Computers & Education*, 138, 13–32. <https://doi.org/10.1016/j.compedu.2019.04.011>
- Selby, C. C., & Woollard, J. (2013). Computational thinking: The developing definition. In *Paper Presented at the 18th annual conference on innovation and Technology in Computer Science Education*. Canterbury. Retrieved from <https://eprints.soton.ac.uk/356481/>.
- Stanovich, K. E. (1986). Matthew effects in reading: Some consequences of individual differences in the acquisition of literacy. *Reading Research Quarterly*, 21(4), 360–407. <https://doi.org/10.1598/RRQ.21.4.1>
- Stanovich, K. E. (2000). *Progress in understanding reading: Scientific foundations and new frontiers*. Guilford Press.
- Strawhacker, A. L., & Bers, M. U. (2015). "I want my robot to look for food": Comparing children's programming comprehension using tangible, graphical, and hybrid user interfaces. *International Journal of Technology and Design Education*, 25(3), 293–319. <https://doi.org/10.1007/s10798-014-9287-7>
- Strawhacker, A. L., Lee, M. C., & Bers, M. U. (2017). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*. <https://doi.org/10.1007/s10798-017-9400-9>
- Strawhacker, A., & Bers, M. U. (2019). What They Learn when They Learn Coding: Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research & Development*, 67(3), 541–575. <https://doi.org/10.1007/s11423-018-9622-x>
- Sullivan, A., & Bers, M. U. (2015). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*.
- Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO Robot Demo: Engaging young children in programming and engineering. Medford, MA, June 21–25. In *Proceedings of the 14th international conference on interaction design and children (IDC '15)*. New York, NY: ACM. Retrieved from <https://sites.tufts.edu/devtech/files/2018/02/IDC-KIBO-Demo-Complete.pdf>.
- Sullivan, A., Strawhacker, A., & Bers, M. U. (2017). Dancing, drawing, and dramatic robots: Integrating robotics and the arts to teach foundational STEAM concepts to young children. In M. S. Khine (Ed.), *Robotics in STEM education: Redesigning the learning experience*. (pp. 231–260). Springer Publishing.
- Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers in Education*, 148 (103798). <https://doi.org/10.1016/j.compedu.2019.103798>
- Thies, R., & Vahrenhold, J. (2012). Reflections on outreach programs in CS classes: Learning objectives for "unplugged" activities. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 487–492). [https://doi.org/10.1007/978-3-319-98355-4\\_29](https://doi.org/10.1007/978-3-319-98355-4_29)
- Thies, R., & Vahrenhold, J. (2013). *On plugging unplugged into CS classes*. <https://doi.org/10.1145/2445196.2445303>
- Virginia Department of Education. (2017). *VDOE: Computer science standards of learning*. retrieved from [http://www.doe.virginia.gov/testing/sol/standards\\_docs/computer-science/index.shtml](http://www.doe.virginia.gov/testing/sol/standards_docs/computer-science/index.shtml).
- Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. *Learning, Education And Games*, 37. Retrieved from <https://dl.acm.org/citation.cfm?id=2811150>.
- Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. <https://doi.org/10.1145/2157136.2157200>
- Wetzels, R., Matzke, D., Lee, M. D., Roudier, J. N., Iverson, G. J., & Wagenmakers, E. J. (2011). Statistical evidence in experimental psychology: An empirical comparison using 855 t tests. *Perspectives on Psychological Science*, 6(3), 291–298. <https://doi.org/10.1177/1745691611406923>
- Wing, J. (2006). Computational thinking. *CACM*, 49(3), 33–36. <https://doi.org/10.1145/1118178.1118215>. March 2006.
- Wing, J. (2011). *Research notebook: Computational thinking—what and why? The link magazine*. Pittsburgh: Spring. Carnegie Mellon University. Retrieved from <https://www.cs.cmu.edu/link/research-notebookcomputational-thinking-what-and-why>.
- Wohl, B., Porter, B., & Clinch, S. (2015). Teaching computer science to 5–7 year-olds: An initial study with scratch, cubelets and unplugged computing. In *Proceedings of the workshop in primary and secondary computing education* (pp. 55–60).
- Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an emerging competence domain. In *Technical and vocational education and training* (Vol. 23, pp. 1051–1067). [https://doi.org/10.1007/978-3-319-41713-4\\_49](https://doi.org/10.1007/978-3-319-41713-4_49)
- Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2020). Computational thinking test for Beginners: Design and content validation. In *2020 IEEE global engineering education conference (EDUCON)* (pp. 1905–1914). IEEE. <https://doi.org/10.1109/EDUCON45650.2020.9125368>.
- Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers in Education*, 141(103607). <https://doi.org/10.1016/j.compedu.2019.103607>



# TechCheck: Development and Validation of an Unplugged Assessment of Computational Thinking in Early Childhood Education

Emily Relkin<sup>1</sup> · Laura de Ruiter<sup>1</sup> · Marina Umaschi Bers<sup>1</sup>

Published online: 26 May 2020  
© Springer Nature B.V. 2020

## Abstract

There is a need for developmentally appropriate Computational Thinking (CT) assessments that can be implemented in early childhood classrooms. We developed a new instrument called *TechCheck* for assessing CT skills in young children that does not require prior knowledge of computer programming. *TechCheck* is based on developmentally appropriate CT concepts and uses a multiple-choice “unplugged” format that allows it to be administered to whole classes or online settings in under 15 min. This design allows assessment of a broad range of abilities and avoids conflating coding with CT skills. We validated the instrument in a cohort of 5–9-year-old students ( $N = 768$ ) participating in a research study involving a robotics coding curriculum. *TechCheck* showed good reliability and validity according to measures of classical test theory and item response theory. Discrimination between skill levels was adequate. Difficulty was suitable for first graders and low for second graders. The instrument showed differences in performance related to race/ethnicity. *TechCheck* scores correlated moderately with a previously validated CT assessment tool (*TACTIC-KIBO*). Overall, *TechCheck* has good psychometric properties, is easy to administer and score, and discriminates between children of different CT abilities. Implications, limitations, and directions for future work are discussed.

**Keywords** Computational thinking · Assessment · Unplugged · Educational technology · Elementary education

## Introduction

Children need to be computer-literate to be able to fully participate in today’s computer-based society—be it as users or creators of digital technology. Educators, researchers, and policy makers in the USA are recognizing the need to give children access to computer science (CS) education from an early age (Barron et al. 2011; Bers and Sullivan 2019; Code.org 2019; White House 2016). In recent years, efforts have shifted away from teaching children only specific CS concepts and programming skills towards helping them engage with a set of underlying abilities that have been termed computational thinking (CT) skills. CT involves a range of analytical skills that are inherent to the field of CS but applicable to many domains of life, such as thinking recursively, applying abstraction when figuring out a complex task, and using heuristic reasoning to discover a solution (Wing 2006; Wing 2011). Due to the centrality of CT, policy makers are now mandating that early

childhood education include interventions that exercise and develop CT skills (Fayer et al. 2017; US Department of Education, Office of Educational Technology 2017).

Unlike other skills such as language, literacy, or mathematical thinking, there are no valid and reliable assessments to measure young learners’ CT skills. However, assessing CT skills can provide proof of learning and useful feedback for students, educators, and researchers evaluating the efficacy of education programs, curricula, or interventions (K-12 Computer Science Framework Steering Committee 2016; Resnick 2007; Sullivan and Bers 2016).

Despite these recognized benefits, there is currently a lack of validated CT assessments for early elementary school students (Lee et al. 2011). Most CT assessments to date have focused on older children and adults (Fraillon et al. 2018; Román-González et al. 2018; Werner et al. 2012; Chen et al. 2017). Prior work in early age groups involved observational rubrics, interview protocols, project-based coding assessments, or programming language-specific assessments (Bers 2010; Bers et al. 2014; Botički et al. 2018; Mioduser and Levy 2010; Wang et al. 2014). These methods require training the scorers on the evaluation metrics and are often time-intensive, unsuitable for classroom use, and/or require children to be familiar with a specific programming platform (Relkin and Bers 2018).

✉ Emily Relkin  
Emily.relkin@tufts.edu

<sup>1</sup> Eliot-Pearson Department of Child Study and Human Development, Tufts University, 105 College Ave, Medford, MA 02155, USA



To fill this gap and provide an easily administrable, platform-neutral classroom-based CT assessment for young children, we have developed a new instrument called *TechCheck*. *TechCheck* draws upon developmentally appropriate CT concepts and skills (Bers 2018) as well as the underlying principles of CS “unplugged” activities that have been used to teach coding without computers over the past two decades (Bell and Vahrenhold 2018; Rodriguez et al. 2016; [www.code.org](http://www.code.org)). We evaluated *TechCheck* in a large test to answer the following questions:

1. Is *TechCheck* a valid and reliable measure of first and second grade children’s CT skills (i.e., what are *TechCheck*’s psychometric properties)?
2. Can *TechCheck* be readily administered to first and second grade children across multiple early elementary school classrooms?

We will first discuss the concept of CT and provide an operational definition. We then give an overview of existing CT assessments for children and discuss some of their characteristics. We explain the concept of unplugged activities and how it was applied to the creation of the assessment. After describing the content and format of *TechCheck* and its development process, we report the results of the study. The article concludes with a discussion of our findings and future directions.

## Computational Thinking

Seymour Papert (1980) alluded to CT in describing the thought processes of children learning to program in LOGO. Wing (2006) later popularized the idea that CT is a vital skill that is relevant to problem solving in technologic as well as non-technological fields. Wing defined CT as “taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing” (Wing 2008, p. 3717). Although there is increasing recognition of the importance of CT, its conceptual boundaries are still murky. A number of different definitions have been put forward (e.g., Aho 2012; Barr and Stephenson 2011; Cuny et al. 2010; Grover and Pea 2013; Kalelioğlu et al. 2016; Lu and Fletcher 2009; Shute et al. 2017; Wing 2006; Wing 2008; Tang et al. 2020).

Zhang and Nouri (2019) argue that there are three types of CT definitions: generic definitions that focus on the universally applicable skill set involved in solving problems (e.g., Wing 2011; Aho 2012); operational definitions that provide a vocabulary for CT and characterize CT into different sub domains (e.g., CSTA 2011); and educational definitions that provide concepts and competencies (e.g., Brennan and Resnick 2012).

All of these definitions place CT outside of the context of child development. However, when working with young children, CT concepts need to be considered in light of the cognitive and social development that occurs at different ages. Taking a developmental approach, Bers (2018) describes CT in

early education as the ability to abstract computational behaviors and identify bugs (Bers 2018: 70). Bers drew on Papert’s definition of “powerful ideas” as skills within a domain or discipline that are individually meaningful and change how we think or perceive the world and problem solve (Papert 1980). This led to the formation of the “Seven Powerful Ideas” that operationalize CT in terms that are developmentally appropriate and that can be taught through a CS or robotics curriculum explicitly designed for young children (Bers 2018). These Powerful Ideas are as follows: algorithms, modularity, control structures, representation, hardware/software, design process, and debugging. Table 1 provides further definitions for each.

There is a relative paucity of research on CT’s cognitive underpinnings in young children (Kalelioğlu et al. 2016; Yadav et al. 2017). Prior work explored how CT is comprised of several subdomains rather than constituting a unified construct (Grover and Pea 2013; ISTE 2015; Wing 2011). For example, Barr and Stephenson (2011) brought together a group of educators and leaders and proposed that CT embodies nine different subdomains including data collection, data analysis, data representation, problem decomposition, abstraction, algorithm and procedures, automation, parallelization, and simulation. Selby and Woollard (2013) narrowed CT to five subdomains by analyzing prior CT definitions and argued that a CT should be defined as a thought process that involves abstraction, decomposition, algorithmic thinking, evaluation, and generalization. As a consequence, instruments that measure CT skills must probe diverse areas and may not perform as uniformly as standardized tests of other abilities (Román-González et al. 2019). These authors have pointed out that CT is a “liquid term” that represents an approach to problem solving rather than a singular concept. They also extend the concept of CT to include “soft skills” such as “persistence, self-confidence, tolerance to ambiguity, creativity, and teamwork”. As such, most CT assessments are different from standardized tests that focus on more unitary academic skills.

## Assessment of CT in Early Childhood

Over the past two decades, there have been several instruments developed to measure CT skills, but only a small subset of studies focused on CT in young children in early elementary school ages four through nine. Most prior work in early age groups uses interview protocols or project-based coding assessments.

In an interview-based approach, researchers have analyzed the responses that children give during one-on-one interviews as they observe the execution of programming tasks. Mioduser and Levy (2010) showed the outcome of LEGO robotics construction tasks to kindergarteners. The children’s CT level was qualitatively assessed by analyzing the terms that children used to describe the robot’s actions as it navigated through a

**Table 1** Developmentally appropriate powerful ideas of CT (Bers 2018)

Powerful idea	Definition
Algorithms	Sequencing, putting things in order, logical organization
Modularity	Breaking up large tasks into smaller parts, instructions
Control structures	Recognizing patterns and repetition, cause and effect
Representation	Symbolic representation, models
Hardware/software	Recognizing that smart objects are not magical but are human engineered
Design process	Understanding the cyclic nature of creative processes and its six steps, perseverance
Debugging	Identifying and solving problems, developing strategies for making things work, and troubleshooting

constructed environment. For example, children who attributed the robot's actions to magic were given low CT skills ratings and those who provided mechanical explanations were considered more advanced. Wang et al. (2014) used a similar approach with 5-to-9-year-old children, who were asked open-ended questions about a tangible programming task that they created called "T-maze". "T-maze" uses TopCode to convert physical programs into digital code (Horn 2012). The researchers identified elements of CT in the children's responses (e.g., abstraction, decomposition) to determine whether the children grasped these concepts. Bers et al. (2014) analyzed programs created by kindergarteners (ages 4.9 to 6.5 years old) using a tangible and graphical programming language called CHERP. For example, children were tasked with programming their robot to dance the Hokey Pokey. The researchers then assessed four CT concepts by scoring children's projects on a Likert scale. Moore et al. (2020) used task-based interview techniques to assess CT. Three participants were video recorded while they were asked questions and performed tasks using the Code and Go Robot Mouse Coding Activity Set developed by Learning Resources. Researchers explored qualitatively how children use representations and translations to invent strategies for solving problems.

Although interview and project-based assessments provide a window into children's thinking, the format of these assessments and the time they require makes them unsuitable for administration outside of specific research settings. Most specifically, the interview-based approach is both time-consuming and subjective, and may be further limited by the children's capacity to verbalize their thought processes.

Some recent effort has been put into creating CT assessments for young children. Marinus et al. 2018 created the Coding Development (CODE) Test 3–6 (for children between 3 and 6 years of age), which uses the robot Cubetto. CODE requires children to program the robot to go to a specified location on a mat by inserting wooden blocks into a "remote control." The task is to either build the program from scratch or debug an existing program. Children are given maximally three trials to complete each of the 13 items, with more points being awarded if fewer attempts are needed. Although the authors state that CODE is meant to measure CT, their assessment requires coding knowledge raising the possibility that their assessment conflates coding with CT skills.

Relkin and Bers (2019) developed a platform-specific one-on-one instrument called *TACTIC-KIBO*, for children aged 5 to 7 years. *TACTIC-KIBO* involves pre-programmed KIBO robot activities that serve as a basis for the questions and tasks that the child is asked to complete. *TACTIC-KIBO* probes CT skills based on the concepts embodied in the Seven Powerful Ideas described by Bers (2018) (see Table 1). *TACTIC-KIBO* classifies each child in one of four programming proficiency levels derived from the Developmental Model of Coding (Vizner 2017). Scores were highly correlated with expert ratings of children's CT skills, indicating criterion validity. *TACTIC-KIBO* is scored objectively and can be uniformly administered and scored in an average of 16 min. However, like the project-based assessment used by Bers et al. (2014) and the qualitative assessment by Wang et al. (2014), *TACTIC-KIBO* requires that the child has already learned how to use a particular programming platform (in this case, coding associated with the KIBO robot).

Assessments that require prior coding experience are generally unsuitable for use in pre-/post-test designs to evaluate the effectiveness of curricula. Most CT assessments are designed for older children or require skills which are not developmentally appropriate for young children. In addition, there is a risk with assessment of this kind that CT skills may be conflated with coding abilities (Yadav et al. 2017). Research with older children has indicated that coding can become automatic and does not always require thinking computationally (Werner et al. 2014). It is therefore desirable to have methods of measuring CT skills that do not require knowledge of computer programming.

### Unplugged Assessments

Assessments that do not require specific programming knowledge are called "unplugged" assessments. The term comes from activities used in teaching, where educators integrate activities that do not require knowledge of computers or other technologies into the CS curriculum. Such activities are often referred to as "unplugged" to reflect that they do not require electronic technology (Bell and Vahrenhold 2018). Typically, unplugged activities are used to exemplify CT principles and provide a hands-on experience without the use of computers or other technologies. An example of an unplugged activity aligned with the concept of algorithms is having students recount the process of brushing

their teeth. Each of the steps (e.g., finding the toothbrush, finding the toothpaste, wetting the brush, applying toothpaste to the brush) must be identified and applied in a specific sequence. By presenting a readily understood analogy, unplugged activities convey CS concepts without requiring students to have access to a computer or actual computer programming experience.

In recent years, unplugged activities have been used in the context of assessment for pedagogical purposes and more recently applied to the assessment of CT skills, mostly in higher elementary and older school children. [Code.org](http://www.code.org) ([www.code.org](http://www.code.org)) provides a widely used online resource for teaching computer programming to elementary school children in kindergarten to fifth grade (ages four to thirteen). [Code.org](http://www.code.org) uses unplugged activities as assessments in its end-of-lesson quizzes. However, [code.org](http://www.code.org) does not provide a scoring system or basis for interpreting the results of the quizzes, and there is no way to compile results over multiple lessons for summative assessment purposes.

The “Bebras” challenge ([www.bebas.org](http://www.bebas.org)) is a name that is strongly associated with unplugged assessments. It is an international contest for 8-to-18-year-olds, in which participants need to solve tasks that are intended to measure CT skills’ transfer to real-life problems. Participants receive points for solving tasks of varying levels of complexity and difficulty. It is, however, not a validated assessment nor is it suitable for routine classroom use (Dagiene and Stupurienė 2016).

One of the most sophisticated and best validated unplugged assessment tools to date has been created by Román-González et al. 2017 The “Computational Thinking test” (CTt) was designed to identify “computationally talented” children among Spanish middle school students (i.e., 10 to 15 years old). It is a 28-item, multiple-choice test covering the computational concepts of sequences, loops, conditionals, and operators. Each item is one of three tasks types: sequencing (bringing a set of commands in sequence), completion (complete an incomplete given set of commands), or debugging (find and correct an error in a given set of commands). The CTt has been found to correlate with spatial ability, reasoning ability, and problem-solving ability, as well as verbal ability. It is administered online, allowing collective administration, and it is used in pre-/post-test designs (Román-González et al. 2018). Since it was designed for middle school students, it is, however, not developmentally appropriate for use with early elementary school children. It is also worth pointing out that the CTt does not cover all CT domains as described above (Table 1). For example, the test does not include questions on representation or modularization.

To summarize, for older children, the CTt offers a reliable and valid assessment of CT ability that does not require familiarity with a particular technological platform. For younger children, all existing assessments are tied to a particular platform, and they are mostly qualitative in nature.

To fill this gap, we developed *TechCheck*, the first unplugged CT assessment specifically designed for administration

to children between 5 and 9 years of age in a classroom or online setting, regardless of the level of their prior coding experience or exposure to programming platforms.

## Method

### Domains and Content

*TechCheck* has been developed to assess various domains of CT described by Bers’ (2018) as developmentally appropriate for young children (see Table 1): algorithms, modularity, control structures, representation, hardware/software, and debugging, with the exception of the design process. A variety of different tasks are used to probe these domains: sequencing challenges, shortest path puzzles, missing symbol series, object decomposition, obstacle mazes, symbol shape puzzles, identifying technological concepts, and symmetry problems (see Appendix 1). Figure 1 provides an example of a *TechCheck* symmetry problem question designed to probe debugging skills. Although it is one of the Seven Powerful Ideas, design process was not included because it is an iterative and open-ended process with many solutions that cannot be readily assessed with the short multiple-choice format implemented in *TechCheck*.

### Face Validity Process

After developing prototypes for these tasks, we solicited feedback from nineteen evaluators (researchers, CS educators, students) with various levels of expertise in CT to determine whether the questions embodied the domains they were designed to assess, and to evaluate the questions’ appropriateness for the target age groups. Evaluators were given an item and asked to select from four options the one domain that they thought the item was probing. Writing in other domains was also an option. Inter-rater agreement was then assessed. There was an average agreement of 81% among raters. Fleiss’ Kappa indicated consensus among evaluators about the CT domain most associated with each question  $\kappa = 0.63$  (95% CI)  $p < 0.001$ . Although all prototypes were confirmed to probe the intended CT domain, some questions were rejected because their content was judged to fall outside the common knowledge of typical 5-to-9-year-olds. Figure 2 shows two examples of prototype questions that were rejected on those grounds.

### Format and Administration

The current version of *TechCheck* consists of 15 questions presented in a forced-selection multiple-choice format with four options. Responses are given by clicking on one of the four presented options. Each correct response is awarded with one point, with a maximum score of 15 points. Two practice

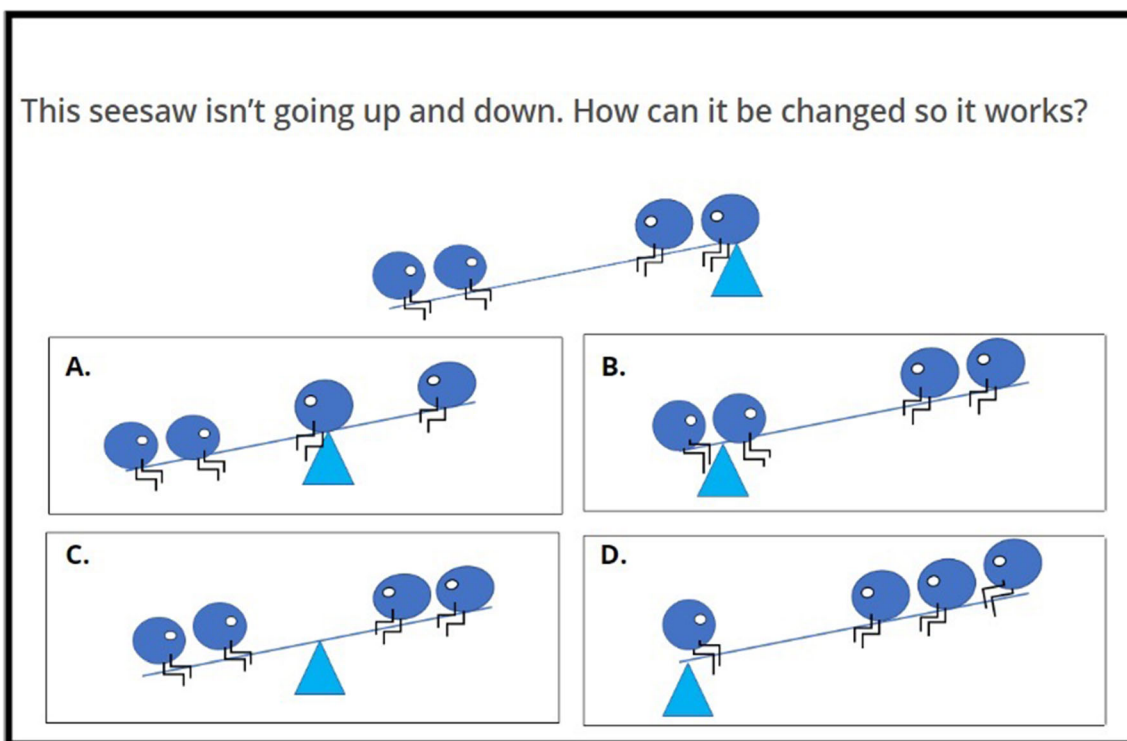


Fig. 1 *TechCheck* symmetry problem question designed to probe debugging skills

questions are included in the beginning of the assessment to familiarize students with the format but are not included in the scoring. All questions must be answered to complete the assessment. *TechCheck* is administered online (currently via a secure online survey platform), which allows it to be administered on multiple platforms including PCs, Android, and Apple devices. The assessment can be administered to children who are pre-literate, and for this reason, administrators are instructed to read each question out loud to the students twice and give them up to 1 min to answer each question.

**Test**

We administered *TechCheck* to determine its psychometric properties in a test involving a total of 768 students from the first and second grades (ages five to nine). The study took place during a period in which students from eight schools

participated in coding, robotics, and literacy curriculum for 2 h per week over 6 weeks (second graders) or 7 weeks (first graders).

**Participants**

Participants were recruited with parental opt-out consent from eight schools in the same school district in Virginia. All schools had a high number of military-connected and low-income students. Owing to absenteeism and other causes, several participants did not complete all scheduled assessments. Altogether, 768 5-to-9-year-olds (mean age 7 years, 6 months) participated. Only participants that were reported to be neurotypical and understood English were included. Figure 3 shows the participant selection diagram indicating how many participants completed *TechCheck* as well as the subgroup that took both *TechCheck* and *TACTIC-KIBO*.

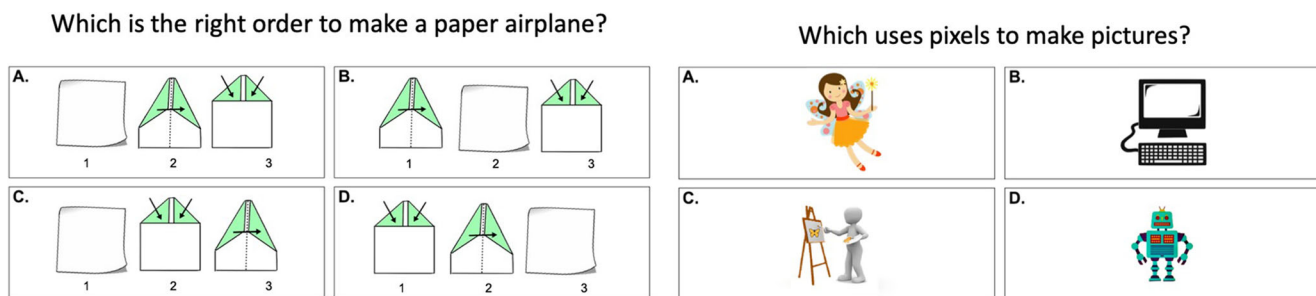


Fig. 2 Examples of two rejected items. These questions fall outside of common knowledge for 4–8-year-olds. Prior knowledge is needed to understand how to make a paper airplane and what pixels are



Table 2 shows the first and second grade students who completed in *TechCheck*, as well as the demographics for the subset of students that completed both *TechCheck* and *TACTIC-KIBO* assessments. As shown in Table 2, the subset of students who completed the required assessments was relatively well-matched to the entire cohort of students.

**Procedure**

Eight proctors (one per school) were trained to administer the assessment in a consistent manner. Proctors first established rapport with children, and then asked children for their assent to participate. *TechCheck* was administered to each class as a group. *TechCheck* was administered up to three times over the course of a 6- to 7-week curriculum for the purpose of a longitudinal analysis, which is part of a different research project. In order to establish criterion validity of *TechCheck*, students were asked to take an updated version of the *TACTIC-KIBO* assessment, a previously validated coding platform-specific CT measure (Relkin and Bers 2019; Relkin 2018). *TACTIC-KIBO* was administered on a tablet on the same week that students completed *TechCheck* for the third time. The updated version of *TACTIC-KIBO* allows for simultaneous administration to a full classroom. *TACTIC-KIBO* probes similar domains of CT as *TechCheck* but requires knowledge of the KIBO programming language (see Fig. 4).

**Data Analysis**

All statistical analyses were conducted in R (Version 3.6.1, R Core Team 2019) using R Studio version 1.2 (RStudio Team 2018). The Item Information and Item Characteristic Curves for *TechCheck* were used to evaluate the difficulty and discrimination power of individual questions and were calculated using the ltm package in R (Rizopoulos 2006). Inter-rater agreement (Fleiss’ Kappa) was conducted using the irr package in R (Gamer et al. 2019). We used Bayesian *t* tests and Bayesian linear regression to examine the effects of gender and race/ethnicity using the BayesFactor R package version 0.9.12-2 (Morey and Rouder 2015). Bayes factors allow determining whether non-significant results support a null hypothesis (e.g., no difference between genders) or whether there is not enough data (Dienes 2014).

**Results**

**Descriptives**

Across all administrations and both grades, the average *TechCheck* score was 10.65 (*SD* = 2.58) out of a possible 15 points. The range was 1–15 points. The average administration

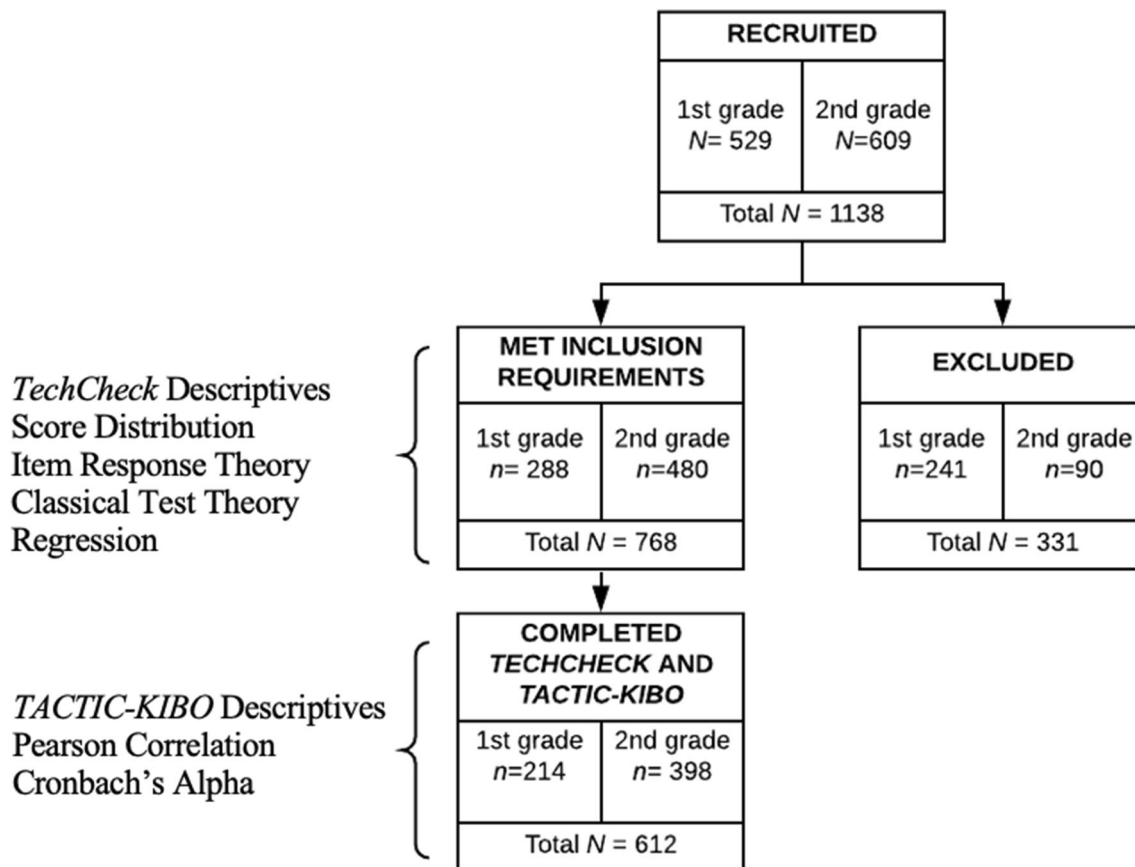


Fig. 3 Participant selection diagram

**Table 2** Demographics of all students who participated in the field test

		Second grade		First grade	
		<i>TechCheck</i> inclusion subgroup	<i>TACTIC-KIBO</i> + <i>TechCheck</i> subgroup	<i>TechCheck</i> inclusion subgroup	Abbreviated <i>TACTIC-KIBO</i> + <i>TechCheck</i> subgroup
Total <i>N</i>		480	398	288	214
Self-reported age	Mean	7.61	7.8	6.23	6.54
	SD	0.58	0.57	0.52	0.62
	Range	6–9	6–9	5–9	5–9
Self-reported gender	Girl	233 (48.54%)	208 (52.26%)	141 (48.96%)	114 (53.27%)
	Boy	243 (50.63%)	177 (44.47%)	144 (50.00%)	94 (43.93%)
	Rather not say	4 (0.83%)	13 (3.27%)	3 (1.04%)	6 (2.80%)
Race/ethnicity	Black/African American	207 (43.13%)	177 (44.47%)	102 (35.42%)	73 (34.11%)
	Hispanic or Latino/a	46 (9.58%)	40 (10.05%)	32 (11.11%)	29 (13.55%)
	Biracial/Multiracial	42 (8.75)	34 (8.54%)	27 (9.38%)	19 (8.88%)
	Asian or Pacific Islander	14 (2.92%)	10 (2.51%)	10 (3.47%)	10 (4.67%)
	White	189 (39.38%)	152 (38.19%)	115 (39.93%)	81 (37.85%)
	American Indian/Native American	4 (0.83%)	1 (0.25%)	2 (0.70%)	2 (0.94%)
	NA	20 (4.17%)	22 (5.53%)	0 (0%)	0 (0%)

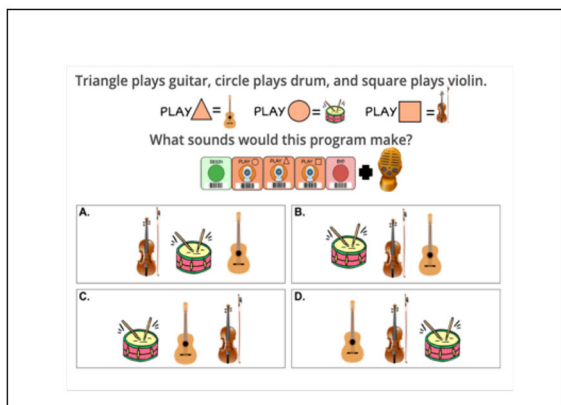
The race/ethnicity “Hispanic or Latino/a” group was not a mutually exclusive category for second graders but was mutually exclusive for first graders due to differences in standardized assessment instruments

time was 13.40 min (*SD* = 05:40). Only 1.50% of all participants scored at or below chance levels (4 or fewer questions correct) and 4.58% answered all questions correctly (see Fig. 5).

Across all administrations in the second grade sample, the mean *TechCheck* score was 11.58 (*SD* = 2.28) out of a possible 15 points. The range was 3–15 points. Administration time averaged 12 min (*SD*, 4.50 min). Across all administrations in the first grade sample, the mean *TechCheck* score was 9.35 (*SD* = 2.39). The range was 1–15 points. Administration time averaged 16 min (*SD* = 5.50 min).

For the second grade subgroup that completed both *TechCheck* and *TACTIC-KIBO* (used to establish criterion validity), the mean *TechCheck* score was 11.86 points (*SD* = 2.37). The mean *TACTIC-KIBO* score was 18.28 points (*SD* = 3.90) out of a possible 28 points. Cronbach’s alpha for *TACTIC-KIBO* was  $\alpha = 0.70$ . The average *TACTIC-KIBO* administration time in this subgroup was 17 min (*SD* = 10 min 32 s).

### TACTIC-KIBO: Representation

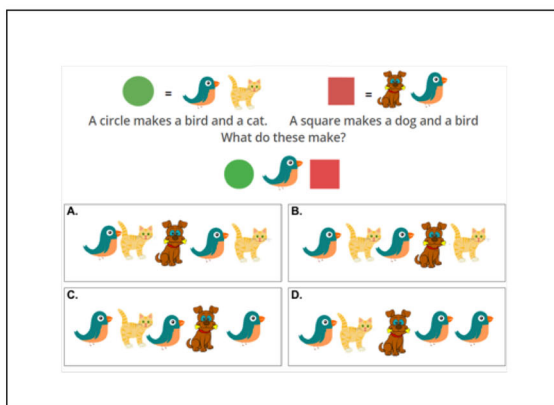


An abbreviated version of *TACTIC-KIBO* was administered to first graders with 21 questions in three different levels of difficulty. The subgroup that completed both *TechCheck* and the abbreviated *TACTIC-KIBO* had an average *TechCheck* score of 9.84 (*SD* = 2.43). The average *TACTIC-KIBO* score for first graders that took both *TechCheck* and *TACTIC-KIBO* was 13.10 out of a possible 21 points (*SD* = 3.33). Cronbach’s alpha for *TACTIC-KIBO* in this subgroup was  $\alpha = 0.67$ . The average administration time for *TACTIC-KIBO* was 22 min (*SD* = 4 min 33 s).

### Gender Differences

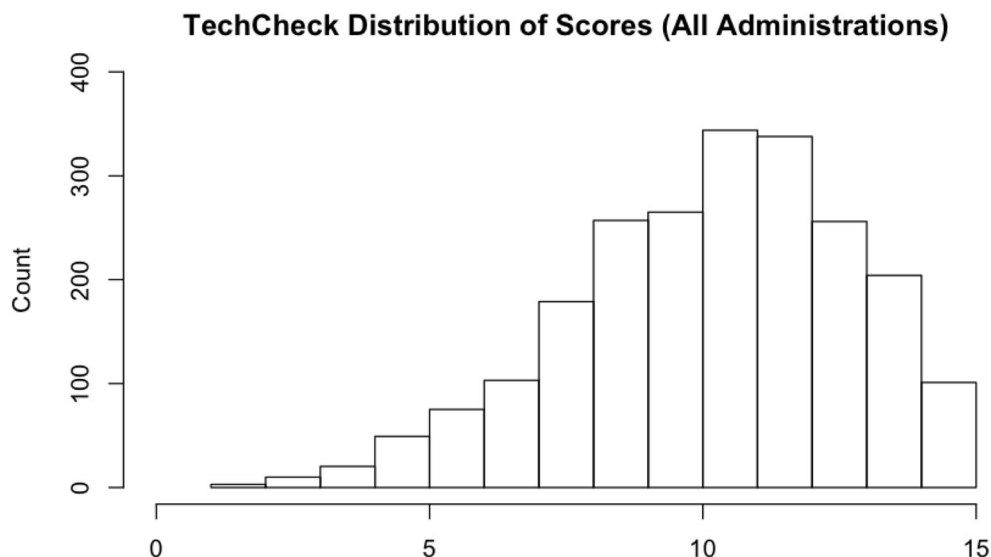
The mean *TechCheck* score for males in the second grade inclusion group was 11.34 points (*SD* = 2.33). The mean for second grade girls was 11.00 points (*SD* = 2.12). There was no statistically significant difference between the two genders

### TechCheck: Representation



**Fig. 4** An example of corresponding types of questions from *TechCheck* and *TACTIC-KIBO* assessment in the CT domain of representation

**Fig. 5** Histogram of the *TechCheck* scores across all administrations ( $N = 2204$ )



( $t = 1.66$ ,  $df = 465.56$ ,  $p > .05$ ). The Bayes factor of 0.38 suggests “anecdotal evidence” of there being no difference between genders (adapted from Jeffreys 1961, cited in Wetzels et al. 2011). Likewise there was no significant difference by gender in first graders, in whom the mean score for boys was 8.66 ( $SD = 2.29$ ) and the mean score for girls was 8.69 ( $SD = 2.35$ ) ( $t = .05$ ,  $df = 282.42$ ,  $p > .05$ ). The Bayes factor of 0.13 suggests “substantial evidence” that there is no difference between genders. Figure 6 shows the distribution of scores of males vs. females in both grades.

### Differences by Racial/Ethnic Background

In the inclusion subgroups, the mean score for Black/African American was 10.21 ( $SD = 2.18$ ) in second grade and 7.94 ( $SD = 2.05$ ) for first grade. Asian/Pacific Islander second grade students had a mean of 11.29 ( $SD = 2.33$ ) and first graders scored on average 11.29 ( $SD = 2.33$ ). White students had an average of 12.21 ( $SD = 1.72$ ) in second grade and an average of 9.27 in first grade ( $SD = 2.4$ ). Latino/a second grade students had a mean of 11.20 ( $SD = 1.98$ ) and first graders had a mean of 8.56 ( $SD = 2.14$ ). Students belonging to other ethnicities/races had a mean of 11.06 ( $SD = 2.36$ ) in second grade and 8.93 ( $SD = 2.66$ ) in first grade. Figure 7 shows the mean scores by race/ethnicity by grade.

A one-way ANOVA examining *TechCheck* scores by race/ethnicity for second graders showed a highly significant difference between groups ( $F(5, 467) = 20.60$ ,  $p < .001$ ). Post hoc analysis (Tukey’s HSD) showed significant differences between Whites and Black/African Americans ( $p < .001$ ) as well as between Whites and biracial/multiracial groups ( $p < .01$ ). A one-way ANOVA examining *TechCheck* scores by race/ethnicity for first graders also showed a significant difference between groups ( $F(5, 282) = 19.78$ ,  $p < .01$ ). Post hoc

analysis (Tukey’s HSD) showed significant differences between Whites and Black/African Americans ( $p < .001$ ).

### Multivariate Modeling

Multiple regression models including gender and race/ethnicity as predictor variables of *TechCheck* score were significant in first graders ( $p < .001$ ) and second graders ( $p < .001$ ). Race/ethnicity was a significant predictor in both of these models (first grade:  $\beta = .25$ ,  $p < .001$ ; second grade:  $\beta = .50$ ,  $p < .001$ ). Gender was not a significant predictor for either grade. Using Bayesian linear regression, we found a Bayes factor of 5.93 for first graders and 6.98 for second graders indicating that the models provided “substantial evidence” that race/ethnicity contributes to the variation in total scores (adapted from Jeffreys 1961, cited in Wetzels et al. 2011).

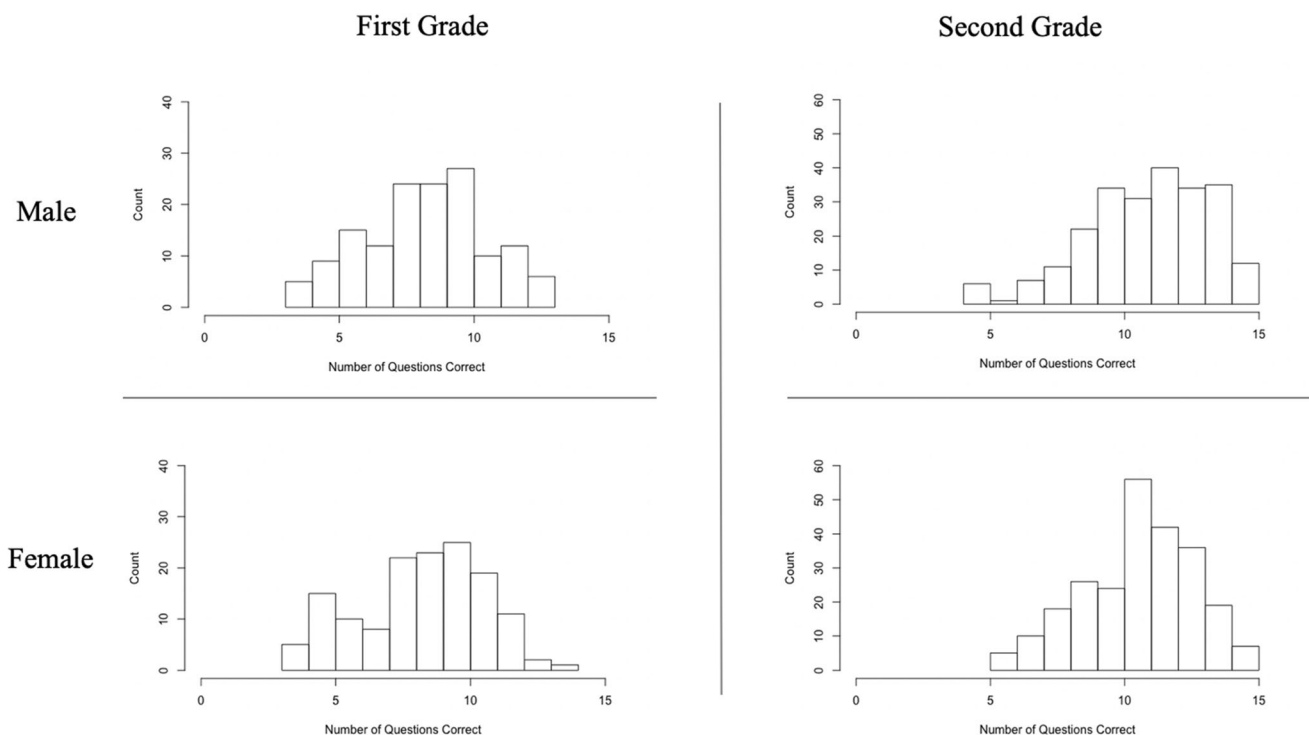
### Criterion Validity

To establish criterion validity, *TechCheck* scores were correlated with scores on an updated version of the *TACTIC-KIBO* assessment (Relkin and Bers 2019; Relkin 2018). *TACTIC-KIBO* is a previously validated assessment that includes 28 questions in four different levels of difficulty.

The association between *TACTIC-KIBO* and *TechCheck* is shown graphically in Fig. 8. A linear correlation is evident but noisy, particularly at lower scores on the two measures. The correlation was moderate at  $r = .53$  ( $p < .001$ ).

### Reliability

We used both Classical Test Theory (CTT) and Item Response Theory (IRT) to evaluate *TechCheck*’s reliability. Using the combination of both CTT and IRT has been



**Fig. 6** Histograms of the *TechCheck* scores for males in second grade ( $n = 233$ ), females in second grade participants ( $n = 243$ ); males in first grade ( $n = 144$ ) and females in first grade (number of  $n = 141$ ). Total number of observations  $N = 768$

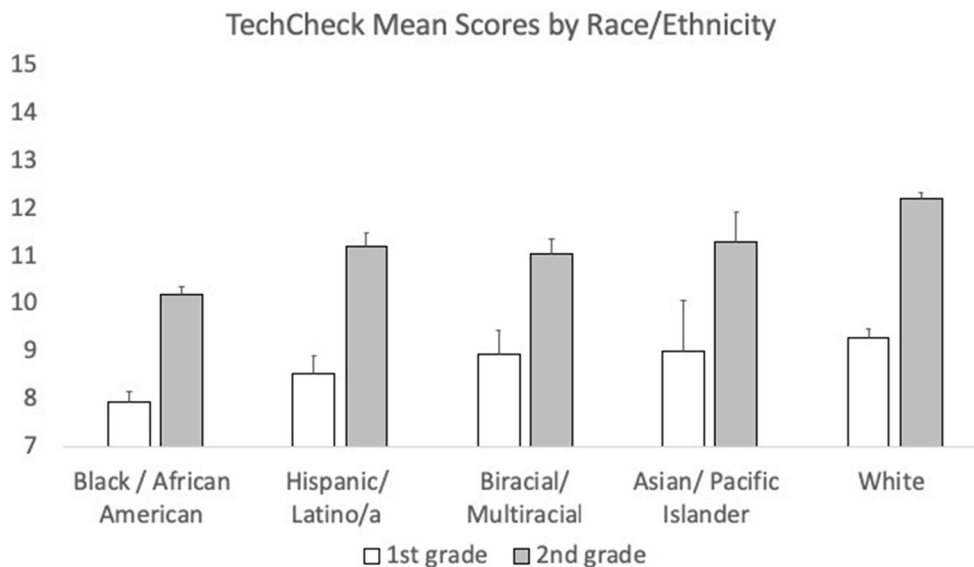
recommended in the context of instrument validation (Embretson and Reise 2000; Cappelleri et al. 2014).

Cronbach’s alpha was calculated as a CTT measure of internal consistency. The observed  $\alpha = 0.68$  is considered moderate to high, and an acceptable level of internal consistency for psychological assessments (Hinton et al. 2004).

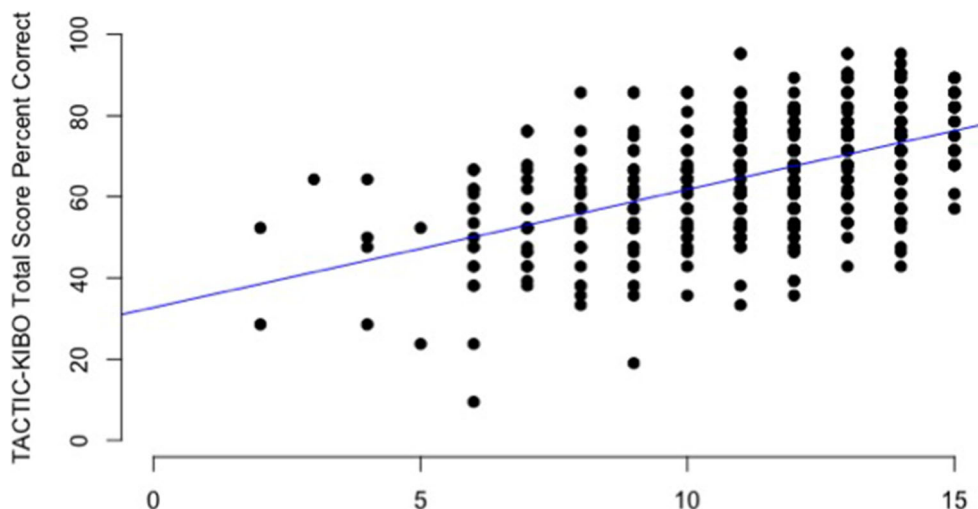
IRT covers a family of models called item response models and the measurement theory that they all have in common (Ramsay and Reynolds 2000). Item response models describe the (mathematical) relationship between

an individual’s response to an item (in our case a *TechCheck* question), and the individual’s underlying trait or ability that the item was intended to measure, in our case CT ability (Kingsbury and Weiss 1983). We used a two-parameter model, which provides information both about the difficulty of each question and its discrimination ability. A questions’ discrimination index indicates how well the question distinguishes between low and high performers. It is desirable to have questions of varying difficulty level and high discrimination.

**Fig. 7** Bar plot showing the mean scores (and standard errors) by race/ethnicity ( $N = 480$  second graders;  $N = 288$  first graders)



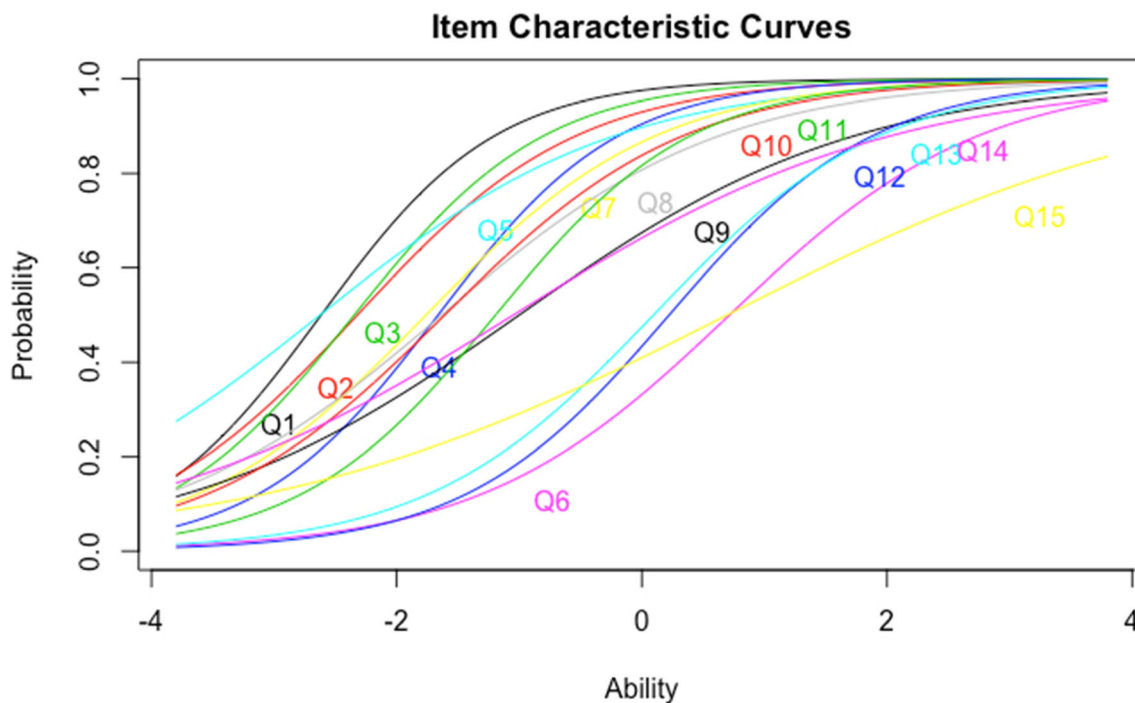
**Fig. 8** Scatterplot showing the relationship between *TACTIC-KIBO* and *TechCheck* ( $N = 612$ )



The IRT analysis results are shown in Fig. 9 (Item Characteristic Curves) and Fig. 10 (Item Information Curves). Item Characteristic Curves (ICC) are S-shaped curves that show the probability of selecting the correct response in *TechCheck* for participants with a given level CT ability. The curves indicate which questions are more difficult and which questions are better at discriminating between students with high and low CT ability. The location of the peak of the curve indicates the level of difficulty, with more difficult questions peaking towards the higher (right) end of the x-axis (ability). The steepness of the curve indicates the question’s discrimination, with steeper curves discriminating better.

The ICCs show peaks at a variety of ability levels, indicating *TechCheck* successfully challenges children with low as well as high CT skill levels. The curves vary in steepness, with all questions showing acceptable levels of discrimination. The mean difficulty index of all items was  $-1.25$  (range =  $-2.63, .7$ ), the mean discrimination index was  $1.03$  (range =  $0.65, 1.41$ ). All indices can be found in Appendix 2.

Item Information Curves (IIC) indicate how much information about the latent ability an item provides. IIC peak at the point at which the item has the highest discrimination. The further the ability levels are away from the peak, the less information can be gained from a particular item for those ability



**Fig. 9** Item Characteristic Curves for all *TechCheck* administrations. The x-axis represents the latent ability of participants, the y-axis the probability of responding correctly to the question ( $N = 2204$ )



levels. In the present sample, most peaks occur either towards the middle or to the left end of the  $x$ -axis (latent ability), indicating that *TechCheck* is better at providing information about students with average or lower CT ability.

## Discussion and Future Directions

CT ability has become a focal point of early Computer Science education. However, up to now, no validated and reliable assessments were available to measure CT ability in younger children who do not have previous coding experience. *TechCheck*, an unplugged assessment in multiple-choice format, was developed to fill this gap. It is designed to be developmentally appropriate for children from kindergarten (age five) through second grade (age nine). In a test with 480 second graders and 288 first graders, we investigated *TechCheck*'s psychometric properties and evaluated whether it could be easily administered across multiple early elementary school classrooms.

Overall, *TechCheck* proved to have moderate to good psychometric properties. This is the first study to show a correlation between the results of an unplugged CT assessment in young children (*TechCheck*) and those obtained using a coding-specific CT instrument (*TACTIC-KIBO*). Their correlation implies that both instruments measure the same underlying ability.

The range of responses was normally distributed without indication of a floor effect. *TechCheck* succeeded in engaging students in a range of CT ability levels. However, a few (less

than 8%) students in second grade received the maximum number of points (15), suggesting that there was a ceiling effect for small number of high-ability children.

The IRT analyses similarly indicated that the level of difficulty was overall appropriate for first and second graders although slightly lower than anticipated. We are planning to assess kindergarten students with *TechCheck* in a future study. Extrapolating from present results, we expect *TechCheck* to perform well in Kindergarten.

Scoring of *TechCheck* was straightforward due to the use of a multiple-choice format and a one point-per-question scoring system. The online platform used in this study did not permit instantaneous reporting of group results upon completion of assessment sessions. We hope to add that functionality in the future. Our platform-specific CT measure, *TACTIC-KIBO*, uses a more complex scoring system that converts raw scores into levels of performance ranging from proto-programmer to fluent-programmer (Relkin 2018). Such a leveling system would not be appropriate for *TechCheck* since it is explicitly not a programming assessment. However, once the data are available for the younger age groups, we plan to develop age-adjusted standards.

The administration of the assessment was feasible in the classroom, and we observed good compliance with the assessment protocol. There were no reported adverse administration experiences. The assessment elicited positive feedback from the teachers and administrators who reported that children were consistently excited to take *TechCheck*. *TechCheck* was successfully administered by multiple proctors working in several classrooms in diverse schools. By these criteria, *TechCheck* demonstrated

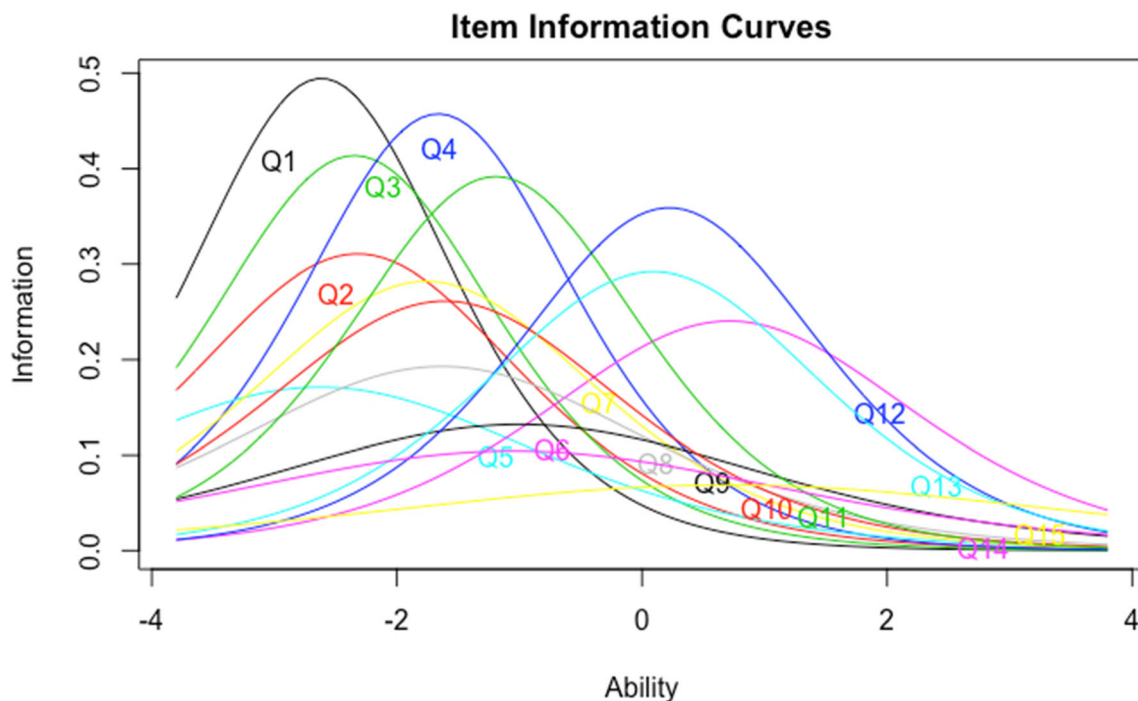


Fig. 10 Item Information Curves for all *TechCheck* administration ( $N = 2204$ )

ease of administration and utility in the setting of early elementary school classrooms.

Overall, *TechCheck* proved to be both a valid and reliable instrument to measure CT ability in young children, and to be readily administered to first and second graders. Román-González et al. (2019) pointed out that CT assessments often focus on concepts rather than “practices and perspectives”, and as a consequence become “static and decontextualized.” Although *TechCheck* is concept- rather than practice-driven, it provides a practical means of assessing CT skills in large numbers of students in a way that correlates with more context-based assessments such as *TACTIC-KIBO*. In the context of education, *TechCheck* may be useful for identifying students with computational talent who can benefit from enriched instruction as well as identifying students with special challenges who require extra support.

To date, there has been no gold standard for measuring CT skills in young children. We used *TACTIC-KIBO* for criterion validation because it was previously validated against experts’ assessments. Some of the data used for this validation was obtained after students were exposed to a KIBO coding curriculum to assure they were familiar with the KIBO coding language. There is a possible bias introduced by exposure to the coding curriculum since students participate in exercises that are somewhat similar to those in the *TACTIC-KIBO* assessment. However, the curriculum did not include unplugged activities of the kind in *TechCheck* making it less likely that bias influenced these results. As additional indicators of criterion validity, the ongoing studies of *TechCheck* with younger children will include measures of standardized mathematical and reading ability (literacy).

Although internal consistency was acceptable from a psychometric testing standpoint, the moderate Cronbach’s alpha raises the issue of whether all questions uniformly measure CT abilities. As discussed in the “[Introduction](#)” section, CT is not a fully unified construct but rather a complex mixture of several domains of thinking and problem-solving abilities.

The fact that CT is not a single unified construct may reduce the internal consistency of any given CT measure. The Computational Thinking test (CTt) for older children (Román-González et al. 2017) has a Cronbach’s alpha of 0.79, which is marginally higher than the 0.68 observed for *TechCheck* in this study. We note, however, that the CTt covers fewer CT domains than *TechCheck*, which is likely to contribute to a higher internal consistency score. *TechCheck* is a composite assessment that probes multiple domains and combines the results into a single total score. *TechCheck* in itself is not designed to quantify CT skills in each of the individual domains it incorporates.

One of the challenges of designing a CT assessment for early elementary school students is variability in reading skills. In the target age group for *TechCheck*, there is typically a combination of literate and emergent-literate children. In this study, proctors read all questions out loud to minimize the

effects of literacy level on the outcome of the assessment. However, it is still expected that literacy level may correlate with CT skills and this represents a potential confounder to this type of analysis. As mentioned above, future studies will also collect measures of children’s literacy skills, allowing us to examine their relationship with *TechCheck* scores to shed light on this question. Alternative methods of administration (e.g., auditory presentation through headphones with an automated proctor) and interactive computerized assessment methods are also worthy of further exploration.

The cohort of this study was ethnically diverse and had a balanced representation of gender. We observed a clear difference in *TechCheck* scores as a function of race/ethnicity. The current study does not allow us to ascertain the basis for the observed difference. Administering *TechCheck* in future studies to students from other backgrounds (e.g., other parts of the USA, other countries) and taking into account socioeconomic and cultural differences as well as students’ performance on other academic measures may shed further light on this issue. Future studies should also explore whether *TechCheck* can be used to accurately assess children who are not typically developing or who are English language learners.

## Conclusions

*TechCheck* has acceptable psychometric properties, is easy to administer and score, and identifies different CT skill levels in young children. *TechCheck* has a suitable design for use in research as well as educational settings. Characterization of *TechCheck*’s utility in longitudinal assessments and in other age groups is currently underway.

**Acknowledgments** We sincerely appreciate our Project Coordinator, Angela de Mik, who made this work possible. We would also like to thank all of the educators, parents, and students who participated in this project, as well as members of the DevTech research group at Tufts University.

**Funding Information** This work was generously supported by the Department of Defense Education Activity (DoDEA) grant entitled “Operation: Break the Code for College and Career Readiness.”






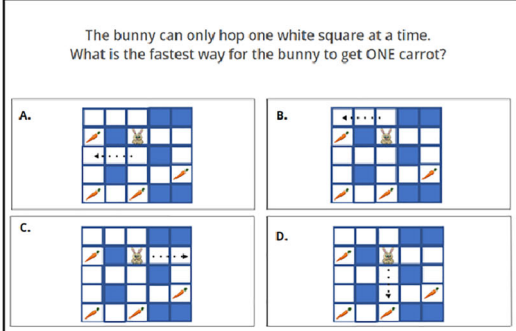
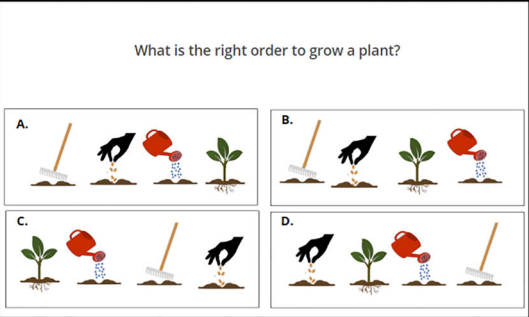
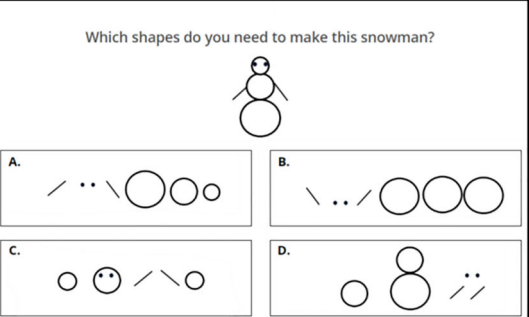



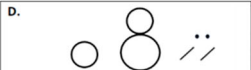
## Compliance with Ethical Standards

**Conflict of Interest** The authors declare that they have no conflict of interest.

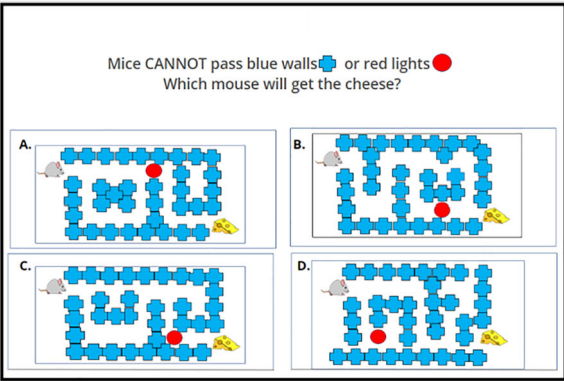
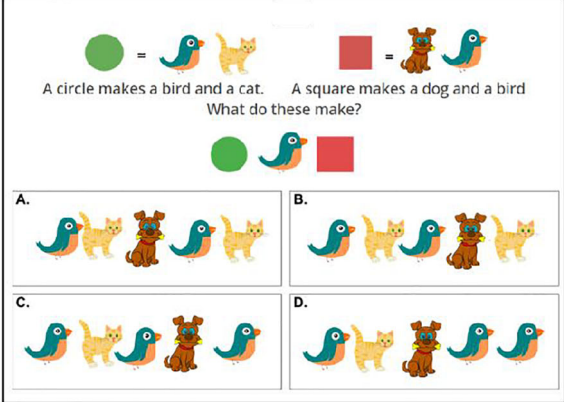
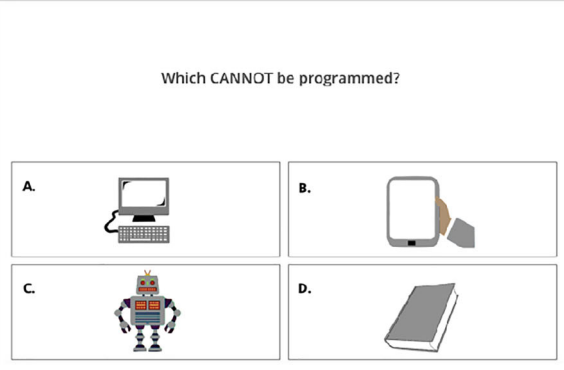
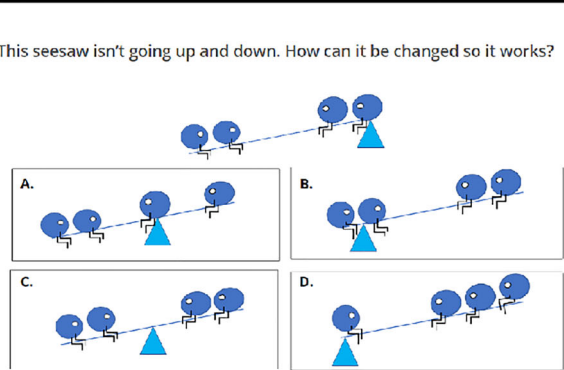
**Ethical Approval** All procedures performed in studies involving human participants were in accordance with the ethical standards of the Tufts University Social, Behavioral & Educational IRB protocol no. 1810044.

**Informed Consent** Informed consent was obtained from the educators and parents/guardians of participating students. The students gave their assent for inclusion.

**Appendix 1. The six CT domains covered in *TechCheck* along with examples of the different tasks used to probe those domains**

CT domain	Task type	<u>Example of a task</u>
Algorithms	Missing symbol series	<p style="text-align: center;">What comes next?</p>  <div style="display: flex; flex-wrap: wrap;"> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">A. </div> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">B. </div> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">C. </div> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">D. </div> </div>
Algorithms	Shortest path puzzles	<p style="text-align: center;">The bunny can only hop one white square at a time. What is the fastest way for the bunny to get ONE carrot?</p> 
Algorithms	Sequencing challenge	<p style="text-align: center;">What is the right order to grow a plant?</p> 
Modularity	Object decomposition	<p style="text-align: center;">Which shapes do you need to make this snowman?</p>  <div style="display: flex; flex-wrap: wrap;"> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">A. </div> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">B. </div> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">C. </div> <div style="width: 50%; border: 1px solid black; padding: 5px; margin: 5px;">D. </div> </div>



<p>Control Structures</p>	<p>Obstacle mazes</p>	<p>Mice CANNOT pass blue walls or red lights</p> <p>Which mouse will get the cheese?</p> 
<p>Representation</p>	<p>Symbol shape puzzles</p>	<p>A circle makes a bird and a cat. A square makes a dog and a bird</p> <p>What do these make?</p> 
<p>Hardware/software</p>	<p>Identifying technological concepts</p>	<p>Which CANNOT be programmed?</p> 
<p>Debugging</p>	<p>Symmetry problem solving</p>	<p>This seesaw isn't going up and down. How can it be changed so it works?</p> 

## Appendix 2. Difficulty and discrimination indexes for the *TechCheck* assessment

Question	Difficulty index	Discrimination index	Question	Difficulty index	Discrimination index
1	-2.62	1.41	9	-1.00	0.73
2	-2.32	1.12	10	-1.61	1.02
3	-2.35	1.29	11	-1.19	1.30
4	-1.67	1.35	12	-0.22	1.20
5	-2.63	0.83	13	-0.094	1.08
6	0.71	0.98	14	-1.05	0.65
7	-1.76	1.06	15	.70	0.525
8	-1.63	0.88			

## References

- Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, 55(7), 832–835. <https://doi.org/10.1093/comjnl/bxs074>.
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *Inroads*, 2(1), 48–54. <https://doi.org/10.1145/1929887.1929905>.
- Barron, B., Cayton-Hodges, G., Bofferding, L., Copple, C., Darling-Hammond, L., & Levine, M. (2011). *Take a giant step: a blueprint for teaching children in a digital age*. New York: The Joan Ganz Cooney Center at Sesame Workshop Retrieved from <https://joanganzcooneycenter.org>.
- Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work?. In *Adventures between lower bounds and higher altitudes* (pp. 497–521). Springer, Cham. [https://doi.org/10.1007/978-3-319-98355-4\\_29](https://doi.org/10.1007/978-3-319-98355-4_29).
- Bers, M. U. (2010). The TangibleK robotics program: applied computational thinking for young children. *Early Childhood Research and Practice*, 12(2) Retrieved from <http://ecrp.uiuc.edu/v12n2/bers.html/>.
- Bers, M. U. (2018). *Coding as a playground: programming and computational thinking in the early childhood classroom*. Routledge. <https://doi.org/10.4324/9781315398945>.
- Bers, M. U., & Sullivan, A. (2019). Computer science education in early childhood: the case of ScratchJr. *Journal of Information Technology Education: Innovations in Practice*, 18, 113–138. <https://doi.org/10.28945/4437>.
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: exploration of an early childhood robotics curriculum. *Computers in Education*, 72, 145–157. <https://doi.org/10.1016/j.compedu.2013.10.020>.
- Botički, I., Kovačević, P., Pivalica, D., & Seow, P. (2018). Identifying patterns in computational thinking problem solving in early primary education. *Proceedings of the 26th International Conference on Computers in Education*. Retrieved from <https://www.bib.irb.hr/950389?rad=950389>
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (Vol. 1, p. 25).
- Cappelleri, J. C., Lundy, J. J., & Hays, R. D. (2014). Overview of classical test theory and item response theory for the quantitative assessment of items in developing patient-reported outcomes measures. *Clinical Therapeutics*, 36(5), 648–666. <https://doi.org/10.1016/j.clinthera.2014.04.006>.
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers in Education*, 109, 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>.
- Code.org (2019). Retrieved from <https://code.org/>
- Core Team, R. (2019). *R: a language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing Retrieved from <https://www.R-project.org/>.
- Computer Science Teachers Association (CSTA) Standards Task Force CSTA K-12 computer science standards (2011), p. 9 Retrieved from: [http://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/CSTA\\_K-12\\_CSS.pdf](http://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf)
- Cuny, J., Snyder, L., & Wing, J.M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress, referenced in <http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>
- Dagiene, V., & Stupurienė, G. (2016). Bebras—a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in education*, 15(1), 25–44. <https://doi.org/10.15388/infedu.2016.02>.
- Dienes, Z. (2014). Using Bayes to get the most out of non-significant results. *Frontiers in Psychology*, 5, 781. <https://doi.org/10.3389/fpsyg.2014.00781>.
- Embretson, S. E., & Reise, S. P. (2000). *Multivariate applications books series*. Item response theory for psychologists. Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers. Retrieved from <https://psycnet.apa.org/record/2000-03918-000>
- Fayer, S., Lacey, A., & Watson, A. (2017). BLS spotlight on statistics: STEM occupations—past, present, and future. Washington, D.C.: U.S. Department of Labor, Bureau of Labor Statistics. Retrieved from <https://www.bls.gov>.
- Frailon, J., Ainley, J., Schulz, W., Duckworth, D., & Friedman, T. (2018). International Computer and Information Literacy Study: ICILS 2018: technical report. Retrieved from <https://www.springer.com/gp/book/9783030193881>

- Gamer, M., Lemon, J., Fellows, I. & Singh, P. (2019) Package ‘irr’. Various coefficients of interrater reliability and agreement. Retrieved from <https://CRAN.R-project.org/package=irr>
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Research*, 42(1), 38–43. <https://doi.org/10.3102/0013189X12463051>.
- Hinton, P., Brownlow, C., McMurray, I., & Cozens, B. (2004). *SPSS explained*. Abingdon-on-Thames: Taylor & Francis. <https://doi.org/10.4324/9780203642597>.
- Horn, M. (2012). TopCode: Tangible Object Placement Codes. Retrieved from: <http://users.eecs.northwestern.edu/~mhorn/topcodes>.
- ISTE. (2015). CT leadership toolkit. Retrieved from <http://www.iste.org/docs/ct-documents/ct-leadershiptoolkit.pdf?sfvrsn=4>
- Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford: Oxford University Press.
- K-12 Computer Science Framework Steering Committee. (2016). K–12 computer science framework. Retrieved from <https://k12cs.org>.
- Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. Retrieved from [https://www.researchgate.net/publication/303943002\\_A\\_Framework\\_for\\_Computational\\_Thinking\\_Based\\_on\\_a\\_Systematic\\_Research\\_Review](https://www.researchgate.net/publication/303943002_A_Framework_for_Computational_Thinking_Based_on_a_Systematic_Research_Review)
- Kingsbury, G. G., & Weiss, D. J. (1983). A comparison of IRT-based adaptive mastery testing and a sequential mastery testing procedure. In *New horizons in testing* (pp. 257–283). Academic Press. <https://doi.org/10.1016/B978-0-12-742780-5.50024-X>.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32–37. <https://doi.org/10.1145/1929887.1929902>.
- Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. In *ACM SIGCSE Bulletin* (Vol. 41, No. 1, pp. 260–264). ACM. <https://doi.org/10.1145/1539024.1508959>.
- Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the cognition of coding in 3-to-6-year olds: the development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language. Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER '18, 133–141. <https://doi.org/10.1145/3230977.3230984>.
- Mioduser, D., & Levy, S. T. (2010). Making sense by building sense: kindergarten children’s construction and understanding of adaptive robot behaviors. *International Journal of Computers for Mathematical Learning*, 15(2), 99–127. <https://doi.org/10.1007/s10758-010-9163-9>.
- Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple representations in computational thinking tasks: a clinical study of second-grade students. *Journal of Science Education and Technology*, 29(1), 19–34. <https://doi.org/10.1007/s10956-020-09812-0>.
- Morey, R. D., & Rouder, J. N. (2015). BayesFactor 0.9. 12-2. Comprehensive R Archive Network Retrieved from <https://cran.r-project.org/web/packages/BayesFactor/index.html>.
- Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books. Retrieved from <https://dl.acm.org/citation.cfm?id=1095592>.
- Ramsay, M. C., & Reynolds, C. R. (2000). Development of a scientific test: a practical guide. *Handbook of psychological assessment*, 21–42. <https://doi.org/10.1016/B978-008043645-6/50080-X>.
- Relkin, E. (2018). Assessing young children’s computational thinking abilities (Master’s thesis). Retrieved from ProQuest Dissertations and Theses database. (UMI No. 10813994).
- Relkin, E., & Bers, M. U. (2019). Designing an assessment of computational thinking abilities for young children. In L. E. Cohen & S. Waite-Stupiansky (Eds.), *STEM for early childhood learners: how science, technology, engineering and mathematics strengthen learning*. New York: Routledge. <https://doi.org/10.4324/9780429453755-5>.
- Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten in *Proceedings of the 6th Conference on Creativity & Cognition* (CC ‘07), pp. 1–6, ACM. <https://doi.org/10.1145/1254960.1254961>.
- Rizopoulos, D. (2006). ltm: an R package for latent variable modelling and item response theory analyses. *Journal of Statistical Software*, 17(5), 1–25. <https://doi.org/10.18637/jss.v017.i05>.
- Rodriguez, B., Rader, C., & Camp, T. (2016). Using student performance to assess CS unplugged activities in a classroom environment. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 95–100). ACM. <https://doi.org/10.1145/2899415.2899465>.
- Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72, 678–691. <https://doi.org/10.1016/j.chb.2016.08.047>.
- Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction*, 18, 47–58. <https://doi.org/10.1016/j.ijcci.2018.06.004>.
- Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining assessment tools for a comprehensive evaluation of computational thinking interventions. In *Computational thinking education* (pp. 79–98). Springer, Singapore. Retrieved from [https://link.springer.com/chapter/10.1007/978-981-13-6528-7\\_6](https://link.springer.com/chapter/10.1007/978-981-13-6528-7_6).
- RStudio Team. (2018). *RStudio: integrated development for R*. Boston: Studio, Inc. Retrieved from <http://www.rstudio.com/>.
- Selby, C. C., & Woollard, J. (2013). Computational thinking: the developing definition. Paper Presented at the 18th annual conference on innovation and Technology in Computer Science Education, Canterbury. Retrieved from <https://eprints.soton.ac.uk/356481/>.
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. <https://doi.org/10.1016/j.edurev.2017.09.003>.
- Sullivan, A., & Bers, M. U. (2016). Girls, boys, and bots: gender differences in young children’s performance on robotics and programming tasks. *Journal of Information Technology Education: Innovations in Practice*, 15, 145–165. <https://doi.org/10.28945/3547>.
- Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: a systematic review of empirical studies. *Computers in Education*, 148, 103798. <https://doi.org/10.1016/j.compedu.2019.103798>.
- U.S. Department of Education, Office of Educational Technology (2017). Reimagining the role of technology in education: 2017 National Education Technology Plan update. Retrieved from <https://tech.ed.gov/teacherprep>.
- Vizner M. Z. (2017). Big robots for little kids: investigating the role of Sale in early childhood robotics kits (Master’s thesis). Available from ProQuest Dissertations and Theses database. (UMI No. 10622097).
- Wang, D., Wang, T., & Liu, Z. (2014). A tangible programming tool for children to cultivate computational thinking [research article]. <https://doi.org/10.1155/2014/428080>.
- Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. <https://doi.org/10.1145/2157136.2157200>.
- Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. Learning,

- Education And Games, 37. Retrieved from <https://dl.acm.org/citation.cfm?id=2811150>.
- Wetzels, R., Matzke, D., Lee, M. D., Rouder, J. N., Iverson, G. J., & Wagenmakers, E. J. (2011). Statistical evidence in experimental psychology: an empirical comparison using 855 t tests. *Perspectives on Psychological Science*, 6(3), 291–298. <https://doi.org/10.1177/1745691611406923>.
- White House. (2016). Educate to innovate. Retrieved from: <https://www.whitehouse.gov/issues/education/k-12/educate-innovate>.
- Wing, J. M. (2006). Computational thinking. *CACM Viewpoint*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>.
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, 366(1881), 3717–3725. <https://doi.org/10.1098/rsta.2008.0118>.
- Wing, J. (2011). Research notebook: computational thinking—What and why? The Link Magazine, Spring. Carnegie Mellon University, Pittsburgh. Retrieved from: <https://www.cs.cmu.edu/link/research-notebookcomputational-thinking-what-and-why>.
- Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an emerging competence domain. In *Technical and vocational education and training* (Vol. 23, pp. 1051–1067). [https://doi.org/10.1007/978-3-319-41713-4\\_49](https://doi.org/10.1007/978-3-319-41713-4_49).
- Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers in Education*, 141, 103607. <https://doi.org/10.1016/j.compedu.2019.103607>.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.