## Abstract:

This study examined 12 preservice teachers' understanding of computational thinking while planning and implementing a computational thinking activity for fifth grade students. The preservice teachers were enrolled in an add-on computer education license that would certify them to teach computer courses in addition to their primary major area (11 elementary education majors, 1 secondary social studies education major). The preservice teachers were asked to develop a 2 h instructional project for fifth grade students to build on the computational thinking concepts learned during the "Hour of Code" activity. Data was collected from preservice teachers' initial proposals, two blog posts, video recordings of in-class discussions, instructional materials, final papers, and a long-term blog post 3 months after the intervention. Results showcased that the process of developing and implementing computational thinking instruction influenced preservice teachers' understanding of computational thinking. The preservice teachers were able to provide basic definitions of computational thinking as a problem-solving strategy and emphasized that learning computational thinking does not require a computer. On the other hand, some preservice teachers had misconceptions about computational thinking, such as defining computational thinking as equal to algorithm design and suggesting trial and error as an approach to computational problem solving. We provide recommendations for teacher educators to use more directed activities to counteract potential misconceptions about computational thinking.

**Keywords:** Algorithms | Computational thinking | Computer science education | Misconception | Problem solving

## Book chapter:

## Introduction

Computers are a ubiquitous part of our society. To further the use and innovation surrounding computer science as it is used in our society, we need to prepare more computer scientists (Emmott & Rison, 2005). In fact, studies have shown that 50% of the jobs needed in the future are computer science related (Code.org, 2016). In President Obama's 2016 State of the Union Address, he placed emphasis on the importance of computer science education when he proposed a new initiative, "Computer Science for All," which focuses on providing students of all ages with access to quality CS education to improve their computational thinking skills for our increasingly computer-focused society (Smith, 2016). Many have acknowledged that this initiative is a milestone for computer science education and computational thinking movement in US K-12 education (Department of Education, 2016).

The Computer Science for All initiative emphasizes the need for more research on defining what computational thinking is, curriculum needs, how it could be integrated into the current education system, and how to prepare teachers for these major changes at the preservice and in-service levels (Smith, 2016). Our research study focuses on the teacher side of those needs and examines how a group of preservice teachers' understanding of computational thinking evolved after developing and implementing a computational thinking instructional project in an elementary school.

## Computational Thinking

Papert and Harel (1991) were the first who coined the term "computational thinking" in their 1991 paper on constructionism. They proposed that computational thinking was a shift on students' thinking by contributing to their mental growth and become producers of knowledge using computing. Computational thinking received broader recognition from Wing (2006) who suggested that computational thinking was a critical twenty-first-century skill comparable to reading or math. Wing described computational thinking as "the thought processes involved in formulating a problem and expressing its solution in a way that a computer—human or machine—can effectively carry out" (p. 33). In other words, computational thinking can prompt critical thinking and problem-solving skills (DeSchryver & Yadav, 2015). Computational thinking can be applied to solve problems that extend beyond computer science (Li & Wang, 2012). Since 2006, the professional and academic computer science communities have attempted to define computational thinking, as well as detail where and how it could be applied. Two organizations involved in supporting computer science education are the Computer Science Teachers Association (CSTA) and International Society for Technology in Education (ISTE). In collaboration with educators and scholars, ISTE and CSTA (Computer Science Teachers Association CSTA, 2011) together provided an operational definition of computational thinking and emphasized computational thinking as a process of formulating and solving problems. Both organizations highlighted the importance of computational thinking as a skill necessary for all students, highlighting that computational thinking was "a problem-solving tool for every classroom" (Phillips, 2009, p. 1). We have summarized computational thinking key tenets as described from a wide range of scholars into six categories: problem solving, decomposition, pattern recognition, abstraction, algorithms, and evaluation.

In most of these aforementioned studies, the researchers have suggested these as potential, but not necessarily required, characteristics to understand and achieve computational thinking skills. Voogt, Fisser, Good, Mishra, and Yadav (2015) suggested that although these characteristics are helpful, they are by no means definitive in defining computational thinking:

> Understanding a concept does not require developing a series of necessary and sufficient conditions that need to be met. In contrast we seek to develop a more graded notion of categories with an emphasis on the possible rather than the necessary. (p. 719)

Therefore, in our study, we do not limit our own definition of computational thinking to these six characteristics as definitive conditions. However, they helped us conceptualize and identify examples of computational thinking in our participants' statements and artifacts.

## Computational Thinking in K12

According to Jonassen (2000), problem solving is the most important cognitive activity that we perform in everyday and professional contexts. He suggested that problem solving could be as simple as how to tie a shoelace or as complex as how to extract protein without lipid contamination. In both situations, we follow a set of specific guidelines, which lead to desirable outcomes. Students in various settings encounter experiences that require problem-solving skills (Hmelo-Silver, 2004). One example of a more specific approach to problem solving can be computational thinking. Many scholars have argued for the inclusion of computational thinking in the K-12 curriculum (Barr & Stephenson, 2011; Lee et al., 2011; Lu & Fletcher, 2009; Sanford & Naidu, 2016; Wing, 2006; Yadav et al., 2014;). For example, Lu and Fletcher (2009) proposed that "teaching students computational thinking early and often…" (p. 261) should be consistently embedded in K-12 teaching activities in order to develop critical thinking and problem-solving skills. Other researchers and educators (Barr & Stephenson, 2011; Lee et al., 2011; Qualls & Sherrell, 2010; Sanford & Naidu, 2016; Yadav et al., 2014) have also provided similar recommendations.

There have been numerous studies at the postsecondary level to examine the affordances of teaching computational thinking in a variety of contexts (Chao, 2016; Cortina, 2007; Li & Wang, 2012; Qin, 2009). For example, in an undergraduate information communication program, 158 students from three classes were introduced to computational thinking during a C++ programming course (Chao, 2016). In this study, Chao found out that programming could prompt students' problem solving "by iteratively formulating diverse programming strategies in a visualized and constructive way" (p. 212). In a different study, Qin (2009) found that 39 students in an undergraduate bioinformatics class were able to use computational thinking abstraction and pattern recognition strategies to solve problems and improve their conceptual understanding of a biology topic. In both of these studies, it is evident that computational thinking could be embedded as part of multiple postsecondary education courses to help future professionals advance their problem- solving skills.

Scholars have suggested that computational thinking can play an important role in the K-12 curriculum (Barr & Stephenson, 2011), suggesting that computational thinking be integrated into other core areas like reading, writing, and mathematics (Sanford & Naidu, 2016). Studies have shown that when computational thinking is integrated into K-12 classrooms, students tend to become motivated from being passive consumers of technology to active contributors, thereby fostering creativity (Voogt et al., 2015). For example, for a high school history class, a student can create an infographic timeline to portray the cause and effect of the

Civil War utilizing decomposition and pattern recognition. The infographic creation process itself is an example of an algorithm progression.

Bers, Flannery, Kazakoff, and Sullivan (2014) conducted a study on 53 kindergarten students employing the "TangibleK" curriculum. TangibleK is a developmentally appropriate technology education curriculum focusing on early childhood robotics and a "learn by doing" approach to programming (Bers et al., 2014). Bers and colleagues found that these young children were intrigued by each other's work, negotiated their ideas to work collaboratively, and were engaged in problemsolving activities. They found that the young children learned computer science, robotics, and computational thinking skills from the TangibleK curriculum. In another study, Lee et al. (2012) used a game play concept to teach 10–15-year-old children computational thinking skills without introducing traditional programming. They designed a system, CTArchade, based on tic-tac-toe and assessed 18 children's conceptual understanding of computational thinking as measured by decomposition, pattern recognition, abstraction, and problem solving or algorithms. The results implied that children were able to articulate more about algorithmic thinking and showed interest in game play after being introduced to the CTArchade system. Both of these studies suggested that benefits of integrating computational thinking into K-12 settings might include dissecting tasks, looking at general patterns, solving problems, and above all fostering critical thinking. However, in an analysis of peer- reviewed empirical 27 articles between 2009 and 2013, Lye and Koh (2014) found that "these studies might not be representative of typical classrooms and these results show that students' learning from computational thinking in naturalistic classroom settings are still not well understood" (p. 57). Currently, teaching computational thinking is typically focused on high school computer science classrooms and some out-of-school environments (Lee et al., 2011). One reason for this may be due, in part, to a lack of teacher knowledge and experience on computational thinking (Kordaki, 2013).

## Preparing Teachers for Computational Thinking

To integrate computational thinking in the K-12 setting, Stephenson, Gal-Ezer, Haberman, and Verno (2005) suggested that teachers need to be included in the process. As with any innovation, teachers have been identified as one of the primary factors that make significant impacts on the successful implementation of innovations. For example, in a study with 25 in-service computer science teachers, Kordaki (2013) found that teachers were critical to the successful implementation of high school computing courses. Therefore, it seems plausible that teachers would also need to be involved in planning for computational thinking in K-12 classrooms. National organizations have recognized that teachers have strong potential influence in the implementation of computational thinking in the classroom (CSTA, 2015). In addition, those same national organizations have provided guidance and resources to teachers in order to embed computational thinking into their curriculum. For example, CSTA and ISTE provide resources on their websites to support the integration of computational thinking into their teaching and learning activities. CSTA has also provided computational thinking standards for kindergarten to twelfth grade. In another example, Google (2015) developed a website that includes lesson plans, videos, and other resources on computational thinking for teachers and administrators in an effort to encourage the integration of computational thinking into all subject areas. Google even started an online certification program for teachers that guide them step by step through the application of computational thinking skills in different subjects, including, but not limited to, math, science, and geography. Other organizations such as Code.org and CS Unplugged have created curriculum focusing on teaching K-12 computational thinking and computer science.

Because scholars have recognized the importance of preparing future teachers to integrate computational thinking into the K-12 classroom, more are conducting studies with preservice teachers to examine their current knowledge and how to best prepare them. For example, Yadav et al. (2014) incorporated one week of computational thinking instruction during an educational psychology course. The 200 preservice teachers had no prior experience with computer science. The instruction focused on how computational thinking ideas could be incorporated into the classroom. The results revealed that the preservice teachers were able to improve their understanding after completing the training module on computational thinking. Yadav et al. (2014) suggested the next step would be to study whether preservice teachers could learn computational thinking by observing teachers as they model related thinking strategies and guide students to use these strategies independently. As suggested by Yadav and colleagues, introducing computational thinking to preservice teachers in their teacher education program may improve their understanding and allow them to see the possible benefits of computational thinking for K-12 students. Furthermore, Yadav et al. stated that "there is very little research on how teachers could be prepared to incorporate [computational thinking] ideas in their own teaching" (p. 13). Therefore, our research question investigated: How do preservice teachers' concepts of computational thinking evolve while planning and implementing a computational thinking activity for fifth grade students?

## Method

This study used a single-case research design to examine preservice teachers' understanding of computational thinking while planning and implementing a computational thinking activity for fifth grade students (Yin, 2013).

## Setting

This case focused on a group of 12 preservice teachers enrolled in an advanced computer education course in a teacher education program at a Midwestern University. All preservice teachers had primary majors in education (11 elementary education majors, 1 secondary social studies education major). The preservice teachers were enrolled in the course to pursue an add-on computer education license, which would certify them to also teach computer applications and computer science, in addition to their primary major area. This course focused on the following concepts: HTML, CSS, algorithms, introductory programming with robotics kits (LEGO Mindstorm, DashDot, Littlebit, Ozobot, Kibo, Sphero), and K-12 programming environments (Scratch, Scratch jr, Code.org). Instruction on computational thinking was embedded within the introductory programming robotics kits and the K-12 programming environments. Preservice teachers were asked to create a video development project to define and describe algorithms. The instruction on computational thinking was covered in the last 6 weeks of the course in the fall of 2015. In addition to this instruction, all 12 preservice teachers read the definitions on the Google (2015) Computational Thinking Certification program online. For the final course project, the preservice teachers developed a 2 h instructional project for fifth grade students to build on the computational thinking concepts learned in the "Hour of Code" activity. The preservice teachers titled this instructional project “Two Hours of Code” and delivered the instruction to a group of 120 students in a public elementary school. The entire project was worth 20% of their final grade in the course. For the “Two Hours of Code” project, the preservice teachers were challenged to meet at least one of the Computer Science Teacher Association (CSTA) standards on computational thinking for grades 3–6. There were three steps to this project: (1) initial proposal development in small group, (2) fifth grade teacher proposal feedback and selection, (3) final proposal revised and developed with all the preservice teachers’ collaboration in the class, and (4) implementation of the selected proposal.

### *Initial Proposal*

Preservice teachers were initially grouped into four groups of three people. Each group had one week to work on a proposal for teaching computational thinking to the fifth grade students. The proposal described an instructional activity that would teach computational thinking. In addition, the proposal needed to explain how the activity would address the CSTA computational thinking standard(s). Each group recorded a video presenting their proposal to the fifth grade teachers.

### *Fifth Grade Teacher Proposal Selection*

One of the fifth grade teachers, who had several years of experience with teaching the “Hour of Code,” reviewed all four groups’ proposals and selected the best option(s). He selected one activity (Teacherbot) from Group 1 and one activity (Scratch Maze) from Group 2 and suggested combining these two proposals. He provided feedback on instructional ideas and conceptual understanding of computational thinking.

### *Fully Developed Instructional Plan*

Next, all the preservice teachers as a class proceeded to develop the computational thinking activity, including manipulatives and resources. The preservice teachers received 3 weeks (six 90 min class sessions) to complete the fully developed proposal. The fully developed proposal included two activities: TeacherBot and Scratch Maze activity. In TeacherBot, students wrote down instructions to navigate their teachers from one spot in the room to another, working around obstacles. In the Scratch Maze activity, students were tasked with creating their own maze in Scratch. After students had created their maze, students created code to navigate their sprite through their maze.

### *Implementation of Selected Proposal*

All preservice teachers were paired up and implemented the two activities into a different fifth grade classroom. Each fifth grade classroom had approximately 20 students. Two preservice teachers taught each group of 20 students.

## Data Collection

Seven data sources were used to document the computational thinking skills of the preservice teachers. The data sources are discussed in chronological order.

***The Initial Group Proposal***

There were four groups that submitted initial group proposals. In these proposals, groups described their suggested potential activities in great detail. They mentioned concepts of computational thinking and pedagogical approaches. They videotaped a presentation of their proposal and sent it to a fifth grade teacher:

- Group 1 (Preservice Teacher A, B, C)
- Group 2 (Preservice Teacher D, E, F)
- Group 3 (Preservice Teacher G, H, I)
- Group 4 (Preservice Teacher J, K, L)

***Pre-blog Reflection Post***

Preservice teachers created a pre-blog reflection post on their proposals. The blog posts ranged from 200 to 500 words. They were asked to reflect on "where is computational thinking in this proposal?," "how would you consider the needs of the teacher and the students in your proposal?," and "how do you find resources and start planning?"

***Video Feedback***

One of the fifth grade teachers provided feedback on the video proposals. The teacher briefly talked about each proposal (strengths and weaknesses). He described the rationale behind the two activities he selected.

***Video Discussion***

After the two activities were selected, the preservice teachers worked together to develop instruction. This collaborative worktime was done during class. This discussion and worktime was videotaped. The worktime lasted 60 min and 75 min in two sessions.

***Post-blog Reflection Post***

Preservice teachers were asked to reflect on their proposal. They were charged with answering questions on their understanding of computational thinking such as "where is computational thinking in the selected proposal?"

***Final Paper***

Preservice teachers were required to complete a final paper. The final paper asked preservice teachers to reflect on their and students' experience on implementing the computational thinking project and answer questions such as "how was the students' experience with computational thinking based on your observations?"

***Long-Term Reflection***

Three months after the class ended, preservice teachers enrolled in a follow-up course (n = 10) were asked to complete a follow-up blog post reflecting on their long-term working memory of computational thinking. Preservice teachers were asked to answer questions about computational thinking such as "how would you explain the concept of computational thinking?"

**Data Analysis**

To analyze the data, we utilized the six characteristics of the framework described in Table 1. After reviewing all the data, two of the researchers went through the first chronological data source (initial group proposal) together. While reviewing this data source, the researchers discussed all codes and found that two additional categories emerged during analysis (definition, misunderstanding): every proposal showed evidence of these additional categories. After the researchers had established a general agreement on the additional categories and practiced coding the first data source together, they divided and coded the remaining data sources separately. Finally, they came together to discuss each coded data. When the researchers' codes disagreed, researchers discussed those instances until agreement was reached (Carey, Morgan, & Oxtoby, 1996). Based on the coded data, the research team reviewed the coded data to establish themes within each code. These are described in the results.

**Table 1. Six categories of computational thinking tenets**

| Characteristic | Definitions (Google, 2015) | Studies that emphasized this characteristic |
|---|---|---|
| Problem solving | Formulating a problem and designing a solution based on the principles of computing | Lu and Fletcher (2009), Wing (2008), Yadav, Mayfield, Zhou, Hambrusch, and Korb (2014) |
| Decomposition | Breaking down data, processes, or problems into smaller, manageable parts | Atmatzidou and Demetriadis (2016), Barr, Harrison, and Conery (2011), Mannila et al. (2014), Qin (2009), Weintrop et al. (2016) |
| Pattern recognition | Observing patterns, trends, and regularities in data | Deschryver and Yadav (2015), Grover and Pea (2013), Peters-Burton, Cleary, and Kitsantas (2015) |
| Abstraction | Identifying the general principles that generate these patterns | Deschryver and Yadav (2015), Grover and Pea (2013), Kramer (2007), Qin (2009), Sanford and Naidu 2016, Wing (2008) |
| Algorithms | Developing the step-by-step instructions for solving this and similar problems | Mannila et al. (2014), Peters-Burton et al. (2015), Wing (2008), Yadav et al. (2014) |

| | | |
|---|---|---|
| Evaluation | Testing and verifying the solution | Atmatzidou and Demetriadis, (2016), Grover and Pea (2013), Peters-Burton et al. (2015), Weintrop et al. (2016) |

**Results**

***Computational Thinking***

In this study, preservice teachers conveyed accurate knowledge of computational thinking: computational thinking does not require a computer, and computational thinking includes elements of problem solving and scaffolding. Although computational thinking emerged from the computer science field, it does not necessarily need to be learned and practiced using a computer (Barr et al., 2011). Preservice teachers were able to represent this knowledge claim. They expressed that computational thinking could exist outside computers and computer science in their initial proposals and maintained this idea through their final papers and 3 months after the course. In all the initial group proposals (which were completed at the beginning of the instructional project), the preservice teachers included instructional activities that did not require a computer to develop fifth grade students' computational thinking skills. One of the groups suggested using pieces of paper to simulate algorithms: "The intro activity [is] an algorithm using computer free-exercises because it only requires the students to move the pieces of paper around to understand the concept" (Group 3, proposal). Another group shared a similar idea where a computer was not necessary: "The snowman activity is computer-free but still shows how algorithms work" (Group 4, proposal). Preservice teachers also expressed the idea of not using computers to teach computational thinking in their pre-blog reflections: "In the warm up, the students have to think about exactly how the teacher can get to the door" (Preservice Teacher B). After the implementation with fifth grade students, the preservice teachers' observations were also aligned with what they planned and then expressed in the final paper:

> During the teacherbot activity, most students physically got up and tried their algorithms out instead of just guessing from their seats. This shows how the students knew how they learned and that acting it out would be more beneficial for them. (Preservice Teacher D)

When asked to define computational thinking in a blog post 3 months after the class, some of the preservice teachers sustained that learning computational thinking can be learned without a computer. For example, one preservice teacher explained how she conducted a computational thinking activity without using computers in a different school environment with third grade students:

> The stop motion dry erase board project I worked on with my 3rd grade class is a really good example. Students look at the problem, think about how they would solve it on paper, write out a plan with a max of 6 slides they draw out to help explain it, think about the way they will word it, and then explain it in a simple yet efficient way to someone who's thinking may not be the same as their [thinking]. (Preservice Teacher G)

**Computational Thinking and Problem Solving**

When asked to define computational thinking, all preservice teachers associated the term with problem solving. For example, in the initial group proposals, one group suggested that students could build a Scratch Maze. When describing their rationale for including this activity, the group stated that "[the students] will also be able to demonstrate their understanding of how the maze and coding can be used to solve a problem" (Group 1), illustrating that they viewed the maze activity as a problem that could be solved using computational thinking skills. Re-blog reflections, which were written after the initial group proposals, also showed the alignment of problem solving and computational thinking: "the concept of computationally thinking has to do with the idea of finding solutions to general, open-ended problems" (Preservice Teacher A). The same theme was observed in the post-blog reflections (e.g., "CT is thinking about how to solve a problem that does not exactly have one answer," Preservice Teacher B) and the final papers (e.g., "Without telling [the 5th graders] how, I guided them in the right direction, and when I came back later they had figured it out and expended upon what I showed them," Preservice Teacher E). These examples highlight the preservice teachers' conceptual understanding of how computational thinking encompasses problem-solving skills. In fact, when asked to define computational thinking 3 months after, all the preservice teachers still expressed problem solving as the purpose of computational thinking. Preservice Teacher I defined it as formulating a problem and solving it: "Computational thinking is the process of thinking and finding solutions to problems." Preservice Teacher G also emphasized problem-solving characteristics of computational thinking: "Computational thinking is the process of thinking and finding solutions to problems." Both these examples show that the preservice teachers sustained the idea that the purpose of computational thinking is problem solving.

## Efficiency in Computational Thinking

Efficiency also emerged from preservice teachers' descriptions. The preservice teachers emphasized that one problem may have multiple solutions and computational thinking could encourage people to find the most efficient solution. This theme was expressed primarily in the initial proposals and post reflections. In the proposals, one group explicitly described efficiency as a requirement in computational thinking: "Each group will be asked to write an algorithm for the teacherbot to use to move through the classroom; however, they are asked to create this using the 'least amount of commands'" (Initial Group Proposal 2). In one preservice teacher's pre-blog reflections, she explicitly shared the requirement for the most efficient solution:

> On a simpler level, when the students create and direct the teacherbot, they have to consider how many times they need the teacherbot to follow the [instructions]. If there is a way to simplify it (loop), they need to solve it by fixing the [instructions]. (Preservice Teacher G)

Although efficiency was only mentioned by seven preservice teachers in the first six data sources, nine preservice teachers mentioned this concept after three months, emphasizing the possibility of multiple solutions and the importance of finding the most efficient one. For example, Preservice Teacher A expressed the possibility of f inding multiple solutions in computational thinking: "Sometimes we can have similar but different step by steps to get the same outcome."

*Characteristics of Computational Thinking*

**Preservice Teachers' Conceptions of Algorithms**

The preservice teachers seemed to present a valid understanding of an algorithm as a set of instructions to complete a task and shared relevant definitions and examples of algorithms throughout the process. For example, groups defined algorithms in their initial proposal as a "set of specific instructions that explains how to complete a task" (Group 4) and provided algorithm examples such as "making a peanut butter and jelly sandwich" (Group 2). In the Initial Group Proposal 4, preservice teachers created an activity that required writing instructions to draw a snowman: "The snowman activity is an algorithm design. Students have to see that they are following a specific set of instructions that gets them to an end of activity." In the Initial Group Proposal 3, preservice teachers designed a robotics activity to teach students algorithms: "Students will be practicing algorithms as well as basic robotics through the cup stacking warm-up activity." These are both relevant examples of algorithms because they focus on step-by-step instructions to complete activities. In the post- blog reflections, preservice teachers placed similar focus on step-by-step instructions in both their definitions and examples. One of the preservice teachers explained how the Teacherbot activity should allow the fifth grade students to create and test their algorithm: "Students have to give the teacher explicit instructions and the teacher has to follow those exact instructions even if they are not correct" (Preservice Teacher D, post-blog reflection).

Even though the preservice teachers had a good understanding of algorithms, they demonstrated a misconception that algorithm design was equivalent to computational thinking. For example, in the initial proposals, Group 2 stated "students will have to use computational thinking to write an algorithm to move their teacherbot around the room." This example shows a misunderstanding that computational thinking is a method for creating an algorithm. Another preservice teacher expressed a similar misconception, equating computational thinking with an algorithm: "Both activities use the idea of creating an algorithm which reflects the idea of computational thinking" (Preservice Teacher F, pre-blog reflection). The algorithm misconception was still prevalent 3 months after the class. Preservice Teacher H related computational thinking to algorithms in his description of computational thinking: "Computational thinking is thinking and working using the same methods that a computer would. For example, solving problems using algorithms, step by step mathematical tools that can lead a computer or an individual to a solution." Although this is not necessarily a comprehensive example, it shows that the preservice teacher may be starting to conceptualize computational thinking as a new type of thinking strategy that could guide them and their students to problem solution and product development.

**Preservice Teachers' Understanding of Pattern Recognition and Abstraction**

After discussing computational thinking with their classmates and implementing the activity in the elementary school, several of the preservice teachers started using other key components of computational thinking, although they did not use the correct vocabulary and explicitly identify those key components (e.g., observing patterns, trends, and regularities and identifying the general principles that generate these patterns). When the preservice teachers were asked to define and connect pattern recognition and abstraction concepts to activities in their assignments, they were not able to identify those and define them explicitly. However, they included examples in the activities. For example, later on in one of the video recordings, they provided an example of abstraction: “We are not going to show them how to make the maze but give them the main components they are going to need to make a maze” (Video Recording 1). Since abstraction is reducing complexity and creating a model of a product, this example shows the preservice teachers’ understanding of abstraction because they were recognizing the different components necessary to build the maze. In another preservice teacher’s post-blog reflections, she described pattern recognition in the Teacherbot activity: “The robot is recognizing a pattern laid out for them and reads the design code sheet to know what movements to make” (Preservice Teacher D). Although some additional characteristics of computational thinking were briefly described by some later in the process, these characteristics were weakly defined.

**Evolution of Trial and Error Approach to Evaluation**

At the beginning strategies of proposal development and the first reflections, the preservice teachers defined a trial and error approach as a strategy to design and test solutions using computational thinking. For example, in the initial proposals, one group suggested using a trial and error approach to test their activities: “trial and error [can] identify what works and does not work in both the Scratch and intro activities” (Group 1). Another group stated using trial and error as a strategy to be used in their activity and real life: “Not only in reference to a corn maze, but students can use this concept to apply the strategy of trial and error in real life situations” (Group 2). These examples show that preservice teachers’ focused on using a trial and error approach, which is inconsistent with computational thinking because it suggested formulation of a problem and a solution. However, as they progressed in the activity design and after implementing it, they had a clearer purpose focusing on an evaluative approach instead, testing the accuracy of their solutions. For example, in the final papers, preservice teachers shared that the students observed each other’s mazes to evaluate their own designs: “[The students] also got to play each other’s mazes and see what different techniques they used” (Preservice Teacher G). After the implementation of the activity, the preservice teachers conf irmed that working in groups and evaluating each other’s mazes were important parts of the activity. One preservice teacher elaborated on the importance of evaluation and how the benefits transferred to product design:

> Another thing that went well was that not all of the teacherbots were perfect algorithms, which allowed for the students to discuss why their algorithms might be off a little. This helped students see how sometimes an algorithm may not work and therefore must be manipulated to work. This was a great way to lead in to Scratch because we let the students know that sometimes their algorithm may not work for scratch and therefore they must change their codes to fix this problem. (Preservice Teacher D, Final Paper)

This example shows that preservice teacher viewed evaluation as an important characteristic of computational thinking process for validating and increasing the efficiency of the students’ solutions.

**Discussion**

We conducted a case study in order to understand how preservice teachers' concept of computational thinking evolved while designing and implementing a computational thinking instructional project. Twelve preservice teachers developed and implemented a 2 h computational thinking instructional project for fifth grade students (n ≈ 125). Results showed that the process of developing and implementing computational thinking instruction seemed to improve preservice teachers' understanding of computational thinking. However, there were still misconceptions of preservice teachers' expressions of computational thinking at the end of the course.

Wing (2006) stated that computational thinking was a necessary skill for K-12 students, similar to reading, writing, and algebra. Computational thinking has been described in a myriad of ways ranging from "an approach to problem solving" (Barr et al., 2011, p. 115) to "an expertise that children are expected to develop" (Grover & Pea, 2013, p. 40). There is not one commonly agreed definition and structure in the literature (Barr & Stephenson, 2011). Our results reaffirm the need for a clearer and consistent definition of computational thinking (Voogt et al., 2015). This lack of consistent definition and understanding creates difficulty when measuring computational thinking (Rich & Langton, 2016). Furthermore, teachers will likely not be able to embed computational thinking in their instruction due to their lack of understanding of the concept (Bower & Falkner, 2015).

In this study, the preservice teachers were able to provide basic definitions of computational thinking as a problem-solving strategy. Efficiency, which was mentioned by some preservice teachers, is commonly presented as one of the criteria for better solutions in problem solving using computational thinking (Weintrop et al., 2016). The preservice teachers also emphasized that learning computational thinking does not require a computer (CSTA, 2011). Computers and programming can make computational thinking stronger conceptually; however, computational thinking without computers is an important unplugged understanding to have as a basic concept before introducing the computer (Lu & Fletcher, 2009). Although the preservice teachers were not able to clearly define or exemplify pattern recognition, decomposition, and abstraction components in their instructional project designs, they were able to successfully define and/or exemplify algorithms and evaluation in the computational thinking process. This understanding of algorithms might be due to their instructor's emphasis on algorithm design in the course and assigning the preservice teachers to create an instructional video about what an algorithm is as a requirement before this project. However, too much emphasis on algorithms seemed to have caused a misconception, causing them to think that algorithms and computational thinking were equivalent. Similar misconceptions about computational thinking have occurred in other studies. In one study, 32 preservice teachers had a misconception that computational thinking was using technology to solve problems (Bower & Falkner, 2015) such as using an office application to complete a task. In another survey study of 200 preservice teachers, Yadav et al. (2014) found that they equated computational thinking with programming. In our study, the preservice teachers interchangeably used algorithms to refer to computational thinking. Even though algorithms are an inevitable part of computational thinking, they are not equivalent to computational thinking (Guzdial, 2010). Algorithms are step-bystep instructions that can help guide us while solving problems; however, computational thinking is more than just algorithm design.

Even though evaluation in computational thinking was not explicitly covered in the course content, the preservice teacher's emphasis on the importance of evaluation in the computational thinking process emerged in their discussions and observations of the fifth grade students’ interactions (testing and validating their solutions with other students in the Teacherbot and Scratch Maze activity). The preservice teachers mentioned trial and error in their initial proposals. However, as they progressed in the process and implemented their instructional project with the f ifth grade students, the term trial and error evolved to a planned evaluation strategy as part of the computational thinking learning goal (Yadav et al., 2014). In real life, people in science fields do not make decision based on a trial and error approach, instead most “analyze a design by creating a model or prototype and collect extensive data on how it performs, including under extreme conditions” (“NSTA,” 2013, p. 9). Hence, we can conclude that developing and implementing an instructional project may have prompted preservice teachers to think more about the process and helped them understand evaluation as a critical characteristic of computational thinking.

## Limitations

The sample size was one limitation of this study as we only focused on 12 preservice teachers from one university. The results would likely change by increasing the sample size or conducting the same research at a different university or classroom context. Furthermore, we did not account for the preservice teachers’ background with computers and assumed all to be similar. Their backgrounds may have made a difference on their conceptions of computational thinking.

The preservice teachers were asked to complete the Google Education’s Computational Thinking Certification (2015) before starting the design and implementation of the instructional project. However, the training website was unavailable for some of the course, and none of the students were able to complete the training. We believe that if they had enrolled and completed the certification, their understanding of computational thinking may have been more defined and comprehensive.

## Conclusion

With its growing importance in K-12 education, it becomes crucial to prepare both preservice and in-service teachers for embedding and implementing computational thinking in their curriculum (Schweingruber, Keller, & Quinn, 2012). This study provided one instance of how computational thinking can progress with preservice teachers as they design computational thinking instruction. In addition, it also helped illuminate where possible misconceptions could exist and what preservice teachers might struggle with in terms of computational thinking. In summary, future work in this area is needed and should include more directed activities to counteract potential misconceptions about computational thinking. Specifically, we suggest that teacher educators work with preservice teachers to help identify the key characteristics such as pattern recognition and abstraction. In our future classes, we intend to use Code.org and CS Unplugged to help further these ideas for our own preservice teachers. Furthermore, we suggest to include computational thinking activities in the current K-12 curricula and prepare preservice teachers from all core subject areas to plan and implement computational thinking as a problem-solving skill in their own classroom (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013).

## References

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661–670.

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. Learning & Leading with Technology, 38(6), 20–23.

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.

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.

Bower, M., & Falkner, K. (2015). Computational thinking, the notional machine, pre-service teachers, and research opportunities. In Proceedings of the 17th Australasian Computing Education Conference, Sydney, Australia.

Carey, J. W., Morgan, M., & Oxtoby, M. J. (1996). Intercoder agreement in analysis of responses to open-ended interview questions: Examples from tuberculosis research. Cultural Anthropology Methods, 8(3), 1–5.

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem- solving through a visual programming environment. Computers & Education, 95, 202–215.

Code.org. (2016). Promote Computer Science. Retrieved from https://code.org/promote.

Cortina, T. J. (2007). An introduction to computer science for non-majors using principles of computation. ACM SIGCSE Bulletin, 39, 218–222.

Computer Science Teachers Association. (2015). CSTA National Secondary School Computer Science Survey. Retrieved from http://www.csta.acm.org/Research/sub/Projects/ResearchFiles/CSTA_NATIONAL_SECONDARY_SCHOOL_CS_SURVEY_2015.pdf.

Computer Science Teachers Association (CSTA). (2011). Operational Definition of Computational Thinking for K–12 Education. Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/ CompThinkingFlyer.pdf.

Department of Education. (2016). Computer Science for All. Retrieved from http://innovation.ed.gov/what-we-do/stem/computer-science-for-all/.

DeSchryver, M. D., & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: Working with teachers to scaffold complex technology-mediated approaches to teaching and learning. Journal of Technology and Teacher Education, 23(3), 411–431.

Emmott, S. & Rison, S. (2005). Towards 2020 Science. Retrieved from http://research.microsoft.com/en-us/um/cambridge/projects/towards2020science/

Google. (2015). Computational Thinking for Educators. Retrieved from https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

Guzdial, M. (2010). Does contextualized computing education help? ACM Inroads, 1(4), 4–6.

Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn?

Educational Psychology Review, 16(3), 235–266.

Jonassen, D. H. (2000). Toward a design theory of problem solving. Educational Technology Research and Development, 48(4), 63–85.

Kordaki, M. (2013). High school computing teachers' beliefs and practices: A case study. Computers & Education, 68, 141–152.

Kramer, J. (2007). Is abstraction the key to computing? Communications of the ACM, 50(4), 36–42.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32–37.

Lee, T. Y., Mauriello, M. L., Ingraham, J., Sopan, A., Ahn, J., & Bederson, B. B. (2012). CTArcade: Learning computational thinking while training virtual characters through game play. In  Proceedings of the Human Factors in Computing Systems Conference, China.

Li, T., & Wang, T. (2012). A Unified approach to teach computational thinking for first year nonCS majors in an introductory course. IERI Procedia, 2, 498–503.

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. ACM SIGCSE Bulletin, 41, 260–264.

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.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in k-9 education. In  Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference (pp. 1–29). New York, NY: ACM.

NSTA. (2013). Next Generation Science Standards. Retrieved from http://www.nextgenscience.org.

Papert, S., & Harel, I. (1991). Situating constructionism. Constructionism, 36, 1–11.

Peters-Burton, E. E., Cleary, T. J., & Kitsantas, A. (2015). The development of computational thinking in the context of science and engineering practices: A self-regulated learning approach. In  Proceedings of the Cognition and Exploratory Learning in the Digital Age Conference, Maynooth, Greater Dublin, Ireland.

Phillips, P. (2009). Computational Thinking: a problem-solving tool for every classroom. Communications of the CSTA, 3(6), 12–16.

Rich, P. J., & Langton, M. B. (2016). Computational Thinking: Toward a unifying definition. In J. M. Spector, D. Ifenthaler, D. G. Sampson, & P. Isaias (Eds.), Competencies in teaching, Learning, and Educational Leadership in the Digital Age: Papers from CELDA 2014. Switzerland: Springer.

Qin, H. (2009). Teaching computational thinking through bioinformatics to biology students. ACM SIGCSE Bulletin, 41(1), 188–191.

Qualls, J. A., & Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum? Journal of Computing Sciences in Colleges, 25(5), 66–71.

Sanford, J. F., & Naidu, J. T. (2016). Computational thinking concepts for grade school. Contemporary Issues in Education Research, 9(1), 23.

Schweingruber, H., Keller, T., & Quinn, H. (2012). A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas. Washington, DC: National Academies Press.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. Education and Information Technologies, 18(2), 351–380.

Smith, M. (2016. Computer Science For All. Retrieved from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.

Stephenson, C., Gal-Ezer, J., Haberman, B., & Verno, A. (2005). The new educational imperative: Improving high school computer science education. In  Final Report of the CSTA Curriculum Improvement Task Force.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies, 20(4), 715–728.

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.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

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.

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.

Yin, R. K. (2013). Case study research: Design and methods. Thousand Oaks, CA: Sage Publications, Inc.