Developing Pre-service Elementary Teacher's Computational Thinking Knowledge Through Coding and Mathematics Pedagogy

Cory Cleasman
Tennessee Tech University, United States
cgleasman@tntech.edu

ChanMin Kim
Pennsylvania State University, United States
chanmin@psu.edu

Abstract: As computer science education standards are disseminated to K-12 school districts nationally, teacher education programs are left with the challenge of ensuring pre-service teachers are prepared to enter their first classroom with the skills and knowledge necessary to align instruction with the new standards. This paper examines the use of a learning intervention called "Block-Based Coding and Computational Thinking for Conceptual Mathematics" (B2C3Math) that aimed to help pre-service teachers majoring in early childhood and elementary education learn and apply computational thinking concepts to their elementary mathematics teaching. Ten pre-service teachers all at the same stage in their teacher preparation program participated in this convergent mixed-methods study. A focus of the research was placed on how participant's computational thinking knowledge changed following the implementation of B2C3Math. Findings suggest that there were changes in the participants' views of computational thinking application to elementary mathematics teaching following the implementation of B2C3Math. Implications for research and instructional practices using B2C3Math for teacher education are discussed.

Introduction

It is recommended state education systems across the United States develop computer science teaching pathways for elementary teachers (Code.org, 2017). Computer science and computer coding are frequently misclassified as the same entity. The study of computer science is much broader and encapsulates theory and concepts around computing. Computer coding can be considered a skill, however, learning any skillful act is reliant upon concepts and knowledge bases to fuel deep learning. Although distinctive terms, educational reform has relied on the use of computer coding as an approach to introduce computer science content to K-12 students (K12 Computer Science Framework Steering Committee, 2016). Coding presented at the K-12 level prepares students for careers requiring even the most basic computer science knowledge due to students being required to understand basic principles such as abstraction, efficiency, and sequencing while learning to code (Barr, & Stephenson, 2011; National Science and Technology Council, 2016). A large portion of various new K-8 computer science standards focus on coding. As educational standards and reform initiatives call for the introduction of computer science through coding, teachers need to be trained to keep up with these demands.

Literature Review

To adequately prepare teachers for 21st-century technological teaching, coding needs to integrate into professional development and more importantly, teacher preparation. Currently, thirty-three states in the U.S. have computer science certifications for high school and middle school teachers (Computer Science Teachers Association, 2018). It is now recommended that elementary teachers rely on professional development to ensure the integration of computer science into curricula (Code.org, 2017). Dismissing the importance of coding at the elementary teacher preparation level is not sensible as coding is a vital 21st-century skill regardless of students' career path choices (Goode, Flapan, & Margolis, 2018; K12 Computer Science Framework Steering Committee, 2016). Pre-service

teachers need training to conceptually integrate computer science instruction through coding, providing them with foundational knowledge and skills making future professional development even more valuable.

Countries and private entities have developed content-specific coding courses and curriculums concentrated on the integration of coding into K-8 education (Falloon, 2016; Moreno-León, Robles, & Román-González, 2016). These types of curriculums, although useful, serve as a drag-and-drop curriculum; frequently leaving elementary teachers without the understanding of how computer science instruction can be incorporated into their already loaded instructional schedule. Practically, integrating coding into disciplines currently being taught allows teachers to more efficiently cover all mandatory content and standards. Coding has been paired with a multitude of elementary disciplines to enhance learning. Historically, dating back to 1980, Seymour Papert's work with LOGO programming was tied to mathematics learning using constructionism learning theory (Papert, 1980). Due to the usability and feasibility of block-based coding for K-12 education, continuing to identify ways in which coding concepts can be crosscut and made more interdisciplinary is vital to its integration into teaching and learning.

Conceptual Framework: Computational Thinking, Coding, and Mathematics

To further emphasize computing concepts, computational thinking has been used as a viable avenue to introduce coding fundamentals and vice versa (Brennan, & Resnick, 2012). For teachers to find computational thinking useful, it needs to be defined and integrated within a specific context and classroom practicality needs to be shown (Barr, & Stephenson, 2011). Within Scratch, programming blocks are color-coded and categorized by specific computational thinking concepts. Brennan and Resnick's (2012) framework advances the Böhm-Jacopini Theorem and identifies seven computational thinking concepts present in coding: "sequences, loops, parallelism, events, conditionals, operators, and data" (p. 3). Coding and computational thinking are concretely interwoven using Brennan and Resnick's (2012) framework.

Computational thinking is required when coding and block-based coding has proven to enhance mathematics teaching and learning (K12 Computer Science Framework Steering Committee, 2016; Lye & Koh, 2014). Furthermore, computational thinking has been identified as a way to connect coding with elementary mathematics for teaching (Gleasman & Kim, 2020). Recently, research has focused on how learners connect computational thinking to mathematics and then how those connections can enhance mathematics education. Middle school students have been shown to struggle with the concept of looping as it relates to mathematics (Grover & Basu, 2017), while Bakos and Thibault (2018) noticed elementary students struggle relating algorithm patterns with number repetition. Benton, Hoyles, & Kalas (2017) further recognized the concept of algorithms as difficult for both students and teachers alike to grasp when connecting mathematics and computational thinking. Pre-service teachers have identified specific computational thinking concepts, such as events and loops, as having more direct connections to elementary mathematics concepts (Gleasman & Kim, 2020). Computational thinking exists outside of both programming and elementary mathematics, however, Brennan and Resnick's (2012) specific computational thinking framework can help bridge the two fields (see Figure 1).

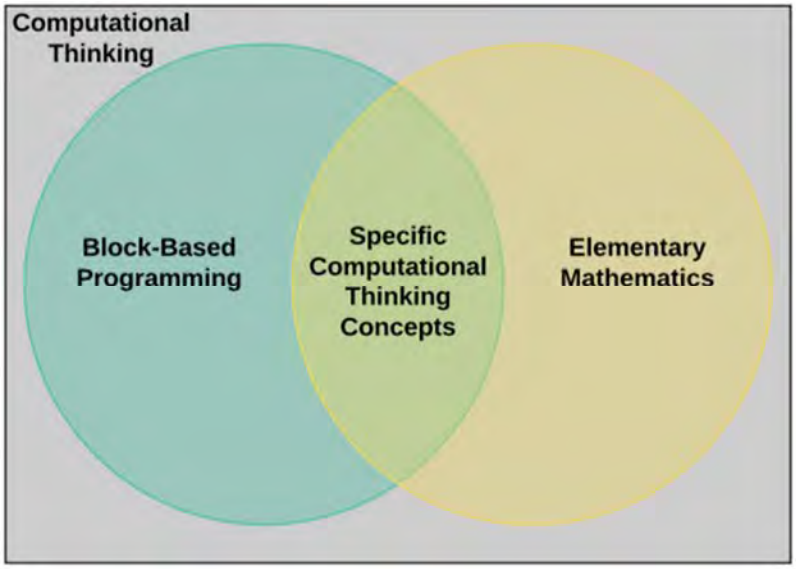

Figure 1. The intersection of elementary mathematics and block-based coding.

Taking into account the relevancy of mathematics and computational thinking connections and the importance of integrating coding into elementary teacher preparation programs, Block-Based Coding and Computational Thinking for Conceptual Mathematics (B2C3Math), was developed to assists pre-service teachers in learning to integrate coding into their K-5 mathematics instruction giving them skills and pedagogical knowledge to adhere to computer science initiatives. K-5 Common Core Mathematics Standards served as the intervention's mathematical content constructs. B2C3Math provides evidence of helping pre-service teachers understand their teaching practice and how computational thinking concepts can be integrated. B2C3Math also provides teacher educators responsible for developing teacher education curriculum with a set of design guidelines to integrate into their teacher preparation instruction (Gleasman & Kim, 2020). Table 1 outlines B2C3Math instruction.

Instructional Weeks	Learning Objective(s)	Instructional Activities
1	Pre-service teachers will be able to understand computational thinking and its application to both mathematics and block-based coding separately. Students will continue to learn the block-based coding language.	-Computational thinking concepts lecture and formative activity. -Block-based coding/computational thinking exploration activity #1
2	Pre-service teachers will investigate elementary mathematics learning opportunities and be able to recognize different instructional methods. Students will continue to learn the block-based coding language.	-Think, pair, share mathematics teaching activity -Block-based coding/computational thinking exploration activity #2
3	Students will be able to proficiently code using the specified block-based programming platform and will be able to identify and extract computational thinking concepts being utilized during coding processes for learning purposes.	-Modeled integrated elementary mathematics and block-based coding lesson -Block-based coding/computational thinking exploration activity #3
4	Pre-service teachers will be able to create an integrated elementary mathematics and coding lesson where learning is transferred through ones understanding of computational thinking.	-Lesson design
5	Pre-service teachers will micro-teach their lesson and receive feedback from peers and instructor.	-Micro-teaching opportunity -Reflective practitioner activity

Table 1. B2C3Math Instructional Outline

Methods

In this section, we describe in detail the process followed for conducting a convergent mixed-methods study. For the purpose of triangulation, both qualitative and quantitative data were gathered and merged, following data collection (Creswell, 2015). Quantitative descriptive statistics were merged with qualitative open-ended survey questions to gain a deeper understanding of the posed research question. Datasets remained separate until analysis, ensuring qualitative and quantitative data did not inform the implementation of the intervention.

Research Question

The purpose of this study was to engage elementary pre-service teachers in B2C3Math and examine how their computational thinking knowledge was altered. The following research question guided the study:

- How did B2C3Math affect elementary pre-service teachers' knowledge and perception of computational thinking?

Participants and Research Context

Ten early childhood and elementary education pre-service teachers attending a large public Southeastern University opted to participate in the Institutional Review Board-approved study. Participants enrolled in a 5-week elective course. The course curriculum consisted of the 5-week intervention module, B2C3Math (see Table 1), which introduced computational thinking as a way to connect elementary mathematics with coding. Each participant was at the same point in their academic coursework and was recruited from a cohort pursuing elementary education bachelor's degrees. Participants had prior experience with block-based coding and robotics through their participation in a National Science Foundation funded project (No. 1712286) and had been enrolled in a mathematics teaching methods course prior to the study. Participants had no formal introduction to computational thinking or how to crosscut coding and mathematics for teaching purposes during their teacher education before this study.

Data Sources

Two data sources were used during this study. The Transformative Robotics Experience for Elementary Students (TREES) assessment (Chen et al., 2017) was used to access participants' understanding of computational thinking, programming structures, and perceptions of programming to teach mathematics. The instrument measures challenges and growth associated with the learning of computational thinking (Chen et al., 2017). Originally intended for the examination of computational thinking learning growth in elementary-aged students, during this study it was used to examine pre-service teachers' learning associated with computational thinking. This instrument was chosen because other computational thinking instruments evaluate computational thinking growth within specific contexts (i.e., Scratch, Alice, etc.) by analyzing products created in these contexts. The TREES assessment stresses the transfer of computational thinking to various contexts without relying on context-specific product analysis. The TREES assessment was used to examine participants' knowledge before and after engaging in B2C3Math.

Yadav et al. (2011) open-ended questions were used to collect qualitative data regarding participants' operational definition of and thoughts about computational thinking and coding within the context of elementary mathematics education. Open-ended survey questions were developed and validated by Yadav et al. (2011).

Data Analysis Methods

Quantitative and qualitative analyses were used to overcome any weaknesses linked to either method (Creswell et al., 2003). Triangulation was implemented to combine the multiple data sources in an effort to increase the validity of results (Greene, 2007). Quantitative findings were enriched and advanced using participants' descriptions and thoughts about computational thinking and coding for the purpose of elementary mathematics

education. Sanitation of data ensured participants' sensitive information was masked from all data sets before analysis.

Descriptive analysis was applied to multiple-choice questions on the TREES instrument datasets investigating percent and mean. A dichotomous scoring scale: 0 for incorrect and 1 for correct, was used to score TREES questions. In order to apply the TREES instrument to the computational thinking concepts used within B2C3Math, each item set was analyzed using Brennan and Resnick's (2012) computational thinking framework. Table 2 lists how Brennan and Resnick's (2012) computational thinking framework concepts relate to specific TREES item sets.

The original TREES instrument consisted of the six sets of assessment items. Due to time constraints, item set 3 on the original instrument was eliminated and the rest of the item sets were renumbered. Item sets 1-2 were themed around everyday scenarios. Item sets 3-5 were focused on robotics activity. An extra code was added to item set 5 directions to provide an additional avenue for participants to use conditionals, a computational thinking concept. The code was "-> if touching [] then []" tells the robot being coded to perform a selection control structure. Please note that the entire instrument is not included. The use of the instrument in this research was granted by Dr. Guanhua Chen.

Item set	Computational Thinking Concept and Programming Structure Utilization (specific item set number)
1	computational thinking: Data (1.1,1.2,1.3, 1.4,), Operators (1.1, 1.2, 1.3, 1.4), Parallelism (1.2), Conditionals (1.3, 1.4)
2	computational thinking: Sequence (2.1, 2.2, 2.3), Conditionals (2.2), Data (2.1, 2.2, 2.3), Operators (2.3)
3	computational thinking: Sequence (3.3), Operators (3.1, 3.2, 3.3), Data (3.1, 3.2, 3.3), Parallelism (3.2, 3.3),
4	computational thinking: Sequence (4.1, 4.2, 4.3, 4.4, 4.5), Loops (4.4, 4.5)
5	computational thinking: Sequence (5.1, 5.2, 5.3, 5.4, 5.5) Event (5.2, 5.3, 5.4, 5.5), Parallelism (5.3, 5.4), Conditional (5.5)

Table 2. TREES Item Sets Associated with Computational Thinking and Programming

Theme identification was used when qualitatively coding open-ended survey responses. Coding nodes within NVivo were developed using relevant research related to computational thinking, mathematics education, and block-based coding. Once one research coded two data sets, a second researcher reviewed the coding to ensure agreement, then the initial researcher coded the rest of the data. Qualitative data were triangulated with the statistical analyses of TREES assessment results.

Findings

We first present findings about changes in overall computational thinking knowledge scores and then place a focus on specific concepts.

Overall Computational Thinking Knowledge Scores

When examining the overall pre- (72%) and post- (76%) percentage score averages of the TREES assessment, it was found that pre-service teacher's computational thinking knowledge increased minimally following B2C3Math implementation. Upon examining percentages on an individual participant basis, the results showed that only two participants scored lower on the post-assessment in contrast to the pre-assessment. However,

when comparing pre- and post-open-ended responses, multiple participants articulate a more comprehensive understanding of what computational thinking is within the B2C3Math context. For example, one student stated on pre-survey, “Computational thinking is placing commands in a specific way into a computer device, so they are understood”. In comparison, the same participant stated on their post-survey: “Computational thinking is to code using specific ideas that apply to a curriculum”. Another participant stated on their post-survey, “Computational thinking is the skills and strategies used in problem solving”. It appears participants may be thinking about computational thinking through a teaching and learning lens compared to her original application of computational thinking to computer devices. A participant sharing a similar sentiment stated “computational thinking is using computer skills” on her pre-survey, while on her post-survey she stated, “Computational thinking is using problem-solving methods that a computer can execute. It is also a way to express problems and find solutions.” Regardless of the TREES assessment score, it seems B2C3Math may have facilitated pre-service teachers to view computational thinking through a lens geared towards teaching and critical thinking, an original goal of the intervention.

The TREES scores of three participants, did not change, while the scores of five participants increased. A 20% overall increase was the largest differential between pre and post TREES scores. Each participant regardless of their computational thinking knowledge growth responded in short form that computational thinking and block-based coding applied to elementary education. However, certain participant responses highlight their perception that the use of computational thinking to teach elementary mathematics with coding may elicit either conceptual or procedural learning. One participant who increased their computational thinking content knowledge following B2C3Math stated, “Students can use computational thinking and coding to show multiplication, division, coordinates, and many more math concepts using procedural knowledge.”. Another participant who had a TREES score which went unchanged stated, “Computational thinking is doing something according to a set of procedures.” A participant who scored lower on the TREES post-assessment stated, “Computational thinking and coding can be used to help students have a deeper understanding of mathematics concepts. They can learn about why an equation works the way it does.” Another participant stated, “Computational thinking is having the ability to think through a concept using the context of coding or mathematical reasoning.” Participants with varying computational thinking knowledge scores seem to have varying views of the instructional methods of coding and computer science integration for mathematics teaching. It is apparent computational thinking knowledge does not affect how teachers view computational thinking as an avenue to facilitate conceptual or procedural mathematics learning, but this difference in viewpoints is apparent. Overall, TREES percentage scores between pre/post assessments changed only slightly, a deeper analysis of valid percentages for specific questions was required.

Question #	Pre- % Correct	Post- % Correct
Q1.1	100	100
Q1.2	100	90
Q1.3	40	40
Q1.4	60	60
Q2.1	100	90
Q2.2	100	100
Q2.3	30	30
Q3.1	90	90
Q3.2: “If we can build all three parts simultaneously, what is the total time needed to build and assemble one machine?” (Shen, 2017, p. 5)	60	90
Q3.3	60	70
Q4.1	100	90
Q4.2	100	100
Q4.3: “Which of the following sequence of these commands, if run by the robotic arm, will produce a rectangle shape on the paper?” (Shen, 2017, p. 6)	80	40
Q4.4: “Now you have a new command “->REPEAT x.” The variable x represents the number of times the previous command will be repeated. What would be the result of running the following code?” (Shen, 2017, p. 7)	20	50
Q4.5	50	60
Q5.1	80	80
Q5.2	90	100

Q5.3	50	70
Q5.4: "A team member wants the robot to move 5 steps to the left, and writes the following code. ->Start. ->Walk 5. [And] ->Turn L. ->End. The code didn't work as expected. Explain why it did not work:" (Shen, 2017, p. 9)	60	100
Q5.5	70	70

Table 3. TREES Pre/Post Assessment Scores by Question #

Comparing pre/post valid percentages for most questions indicates no more than a 20% change (see table 3). In general, this change was positive. Questions 3.2, 4.4, and 5.4 had greater than a 20% increased change. Question 4.3 was the only question resulting in greater than a 20% decrease change between pre- (80%) and post- (40%) scores (see table 3). When investigating changes based on framework-specific computational thinking concepts, a deeper analysis emerged aligned with participant's understanding of specific computational thinking concepts.

Understanding of Specific Computational Thinking Concepts

Pre-service teachers' understanding of events, loops, and parallelism improved substantially. Valid percentages associated with the general computational thinking knowledge scores were useful; however, within the context of B2C3Math, computational thinking was examined through a particular framework (Brennan & Resnick, 2012). To elaborate on how the knowledge of framework-specific computational thinking concepts changed, each TREES assessment question was correlated to the framework (see Table 2). Valid percentages of correct framework-specific computational thinking concepts assessed within pre/post answers for TREES items are noted in Table 4.

Computational Thinking Concepts Assessed	Pre-Score Average	Post-Score Average
Sequence	70.7	78.5
Event	67.5	85
Loops	35	55
Conditionals	67.5	67.5
Parallelism	66	84
Data	74	76
Operators	67.5	71.25

Table 4. Valid Percentages of Computational Thinking Concepts Assessed Correlated to Specific TREES Items

Three participants operationally defined computational thinking listing multiple and/or all computational thinking concepts within their open-ended response. Valid pre/post score averages increased for all but one of the computational thinking concepts, conditionals, which remained the same. The largest positive changes (>17 %) occurred between pre/post scores for the computational thinking concepts events, loops, and parallelism. Valid percentages indicate knowledge of loops and parallelism changed similarly between the administered pre/post TREES assessments. Events resulted in the highest valid percentage post score (85%) of any computational thinking concept assessed. However, loops had the lowest valid percentage post score (55%), but the greatest positive difference between post- and pre-score averages (20%). Participants' knowledge of conditionals went unchanged overall. Following B2C3Math, Participants scored a valid percentage post score of > 70% on questions associated with all computational thinking concepts; except conditionals and loops.

Discussion

Understanding the participants' knowledge of computational thinking can allow for a deep examination of the learning process behind the integration of B2C3Math. It can be acknowledged that B2C3Math had no

statistically significant impact on changing participant computational thinking knowledge, however, participants grasped the knowledge of specific computational thinking concepts more effectively than others after engaging with B2C3Math. Following a deeper investigation into specific questions on the TREES assessment, it was found that pre-service teachers' understanding of events, loops, and parallelism improved. Furthermore, it seems B2C3Math prompted pre-service teachers to start viewing computational thinking as an elementary mathematics teaching and learning tool. This is valuable as effective computer programming teacher education is among the most important factors for ensuring the development of future teacher's positive attitudes towards information and communication technologies (Yildirim, 2000). Their perception of computational thinking was less procedural and more conceptually holistic as a result of B2C3Math instruction. Although not significant, pre-service teachers had a better understanding of all seven computational thinking concepts, except for the conditionals after engaging with B2C3Math. Percentage scores showed pre-service teachers overall had a greater than 70% understanding of all seven computational thinking concepts identified within the TREES assessment, after participating in B2C3Math, except for loops and conditionals. It should be noted that these findings are only based on descriptive analysis and are not statistically significant.

Findings suggest regardless of a positive or negative change in computational thinking knowledge scores, these pre-service teachers viewed computational thinking as an instructional way to connect mathematics and coding for instructional purposes. Furthermore, computational thinking scores do not solely indicate pre-service teachers' perception of computational thinking. However, based on open-ended responses it is apparent there is a differentiation in procedural vs. conceptual teaching methods when it comes to integrating computational thinking, and further research is required.

Implications and Conclusions

There is much work to be done to understand the best avenue of reform for elementary teacher education to keep up with computer science initiatives. As computational thinking has surfaced as a way to understand large computing principles, it is important to relay this educational opportunity to developing teachers. B2C3Math is not an answer, but a step in the right direction. As with any instructional material, iterative revision needs to be made following teacher reflection. Enhancing teachers' computational thinking knowledge and facilitating an investigation of a teaching strategy that does not place more stress on a loaded curriculum but makes teaching interdisciplinary is beneficial for both teacher preparation programs and pre-service teachers. More research is needed to understand how B2C3Math and its corresponding guidelines can be revised to better serve pre-service teachers, as well as, how the learning of computational thinking explicitly relates to mathematical teaching attitudes.

References

Bakos, S., & Thibault, M. (2018). Affordances and tensions in teaching both computational thinking and mathematics. In E. Bergqvist, M. Österholm, C. Granberg, & L. Sumpter (Eds.), Proceedings of the 42nd conference of the International Group for the Psychology of mathematics educations. 2, 107– 114.

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.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. Digital Experiences in Mathematics Education, 3(2), 115–138.

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 (pp.1–25).

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 & Education, 109, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Gleasman, C. & Kim, C. (2020). Pre-service teacher's use of block-based programming and computational thinking to teach elementary mathematics. *Digit Exp Math Education*. https://doi.org/10.1007/s40751-019-00056-1

Code.org. (2017). *Recommendations for states developing computer science teacher pathways*. Retrieved from https://code.org/files/TeacherPathwayRecommendations.pdf

Computer Science Teachers Association. (2018). *State of computer science education*. Retrieved from https://advocacy.code.org/

Creswell, J. W. (2015). *A concise introduction to mixed methods research*. Thousand Oaks, CA: Sage.

Creswell, J. W., Clark, V. L. P., Gutmann, M. L., & Hanson, W. E. (2003). *Advanced mixed method research designs*. In A. Tashakkori & C. Teddlie (Eds.), Handbook of mixed methods in social and behavioral research (pp. 209-240). Thousand Oaks, CA: Sage.

Falloon, G. (2016). An analysis of young students thinking when completing basic coding tasks using Scratch Jnr. on the iPad. Journal of Computer Assisted Learning. 32(6), 576-593.

Goode, J., Flapan, J., & Margolis, J. (2018). Computer science for all: A school reform framework for broadening participation in computing. In W. G. Tierney, Z. B. Corwin, & A. Ochsner (Eds.), Diversifying digital learning: Online literacy and educational opportunity (pp. 45–65). Baltimore, MD: Johns Hopkins University Press.

Greene, J. C. (2007). *Mixed methods in social inquiry*. San Francisco: Jossey-Bass.

Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming. In M. Caspersen, S. Edwards, T. Barnes, & D. Garcia (Eds.), Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education (pp. 267–272). New York, NY: ACM Press.

K12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. Retrieved from http://www.k12cs.org

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

Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the K-12 curriculum? Journal of Information Technology Education: Research, 15, 283–303.

National Science and Technology Council. (2016). *The national artificial intelligence research and development strategic plan*. The National Academies Press.

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

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. In Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (pp. 465–470). New York, NY, USA: ACM. https://doi.org/10.1145/1953163.1953297

Yildirim, S. (2000), Effects of an educational computing course on preservice and inservice teacher: A discussion and analysis of attitudes and use, Journal of Research on Computing in Education, 32(4), 479-496

Acknowledgements

This research is partially supported by grant 1712286 from the National Science Foundation (USA) to PI ChanMin Kim. Any opinions, findings, or conclusions are those of the authors and do not necessarily represent official positions of NSF.