11-26-2020

An Examination of a Group of Middle School Students' Engagement during a Series of Afterschool Computing Activities in an Urban School District

Brendan Calandra
Georgia State University

Maggie Renken
Georgia State University

Jonathan Cohen
Georgia State University

Timothy Hicks
Georgia State University

Tuba Ketenci
Georgia Institute of Technology

Recommended Citation

Calandra, B. Renken, M., Cohen, J., Hicks, T., and Ketenci, T. (2020). An examination of a group middle school students' engagement and computational thinking outcomes during a series of afterschool computing activities. Tech Trends. 65, 17-25. https://doi.org/10.1007/s11528-020-00557-6

An examination of a group middle school students' engagement and computational thinking outcomes during a series of afterschool computing activities

See below for published version:

Calandra, B. Renken, M., Cohen, J., Hicks, T., and Ketenci, T. (2020). An examination of a group middle school students' engagement and computational thinking outcomes during a series of afterschool computing activities. Tech Trends. 65, 17-25. https://doi.org/10.1007/s11528-020-00557-6

Abstract

In this study, the authors analyzed data from a sample of thirty-two middle school students from an inner-city school district in the southeastern United States who used MIT's App Inventor to design, create, and remix mobile apps during an afterschool program for one school year. This paper focuses on computer science learning outcomes associated with computational thinking (CT), and as measured by an instrument created by the authors called the *CT Quiz* (Ayer, et. al. 2018). Findings indicated a linear relation between the number of apps a participant created during the given afterschool program and their level of accuracy on the CT Quiz. Findings should be of interest to both researchers and practitioners.

Keywords: *Computer Science Education, Broadening Participation in Computing, Afterschool, Computational Thinking*

This work was supported by the National Science Foundation grant number 143328.

An examination of middle school student computational thinking outcomes and engagement in a series of afterschool computing activities

Introduction

The study reported in this paper was part of a larger project that included the creation, implementation, and initial testing of an afterschool learning environment for middle school youth in a large, inner-city school district in the southeastern United States. Many young people in the United States do not have equal opportunities to engage in computing, especially girls, minorities, and those from lower-income families (Google & Gallup, 2016; K12CS, 2019), resulting in tech sector jobs and the technologies that those jobs produce in the US still primarily being the effort of a narrow set of demographics that is not representative of the larger population (Wang, 2017). This has in part resulted in a large number of vacant computing jobs in the United States, but also a noticeable lack of diversity in the computing workforce (Code.org, 2019). This lack of diversity can exclude a segment of our population from a chance to share in the socioeconomic stability that high-paid careers can provide. It can also result in new technologies being designed, developed, and implemented in ways that do not equitably serve an entire, diverse population in countries like the US (Howard, 2016; Lee, 2015; Saran, 2017), a phenomenon that can have widespread societal impact. There is a growing body of work focused on broadening participation in computing (Asprey, 2016; Goode, 2008; Lachney, 2018; Rankin & Thomas, 2017; Ryoo, et al., 2013). One way to approach broadening participation reported in this literature is to provide more opportunities for women and people of color to be producers rather than mere consumers of technology in both formal and informal learning environments (Kafai & Burke, 2014). Informal learning environments such as museums, makerspaces, computer labs, community centers, and school-based afterschool programs have been reported in the literature as well-suited for bridging gaps in access to computing education in part because these programs can afford students, some of whom may not have such opportunities during the school day, the time and freedom to learn how to be producers rather than consumers of digital technologies (Ericson & McKlin, 2012; Maloney, et al., 2008; Scott, Sheridan, & Clark, 2015).

The project within which this study was embedded was designed to test a model for afterschool programming that exposes traditionally underrepresented students to accessible, engaging, and challenging computer science curricula. The study described here focuses on one set of student outcomes resulting from participation in the program, and related to a construct called computational thinking (CT). Computational thinking (CT) has been established by a number of educational institutions and government agencies as an essential 21st century skill (e.g., K12CS, 2019). Since Wing's (2006) influential article, quite a few definitions of CT have emerged in the literature. See Grover and Pea for an in-depth analysis (2013). While there are multiple and varied definitions of CT that exist, CT is considered by many as the ability to use principles from computer science to solve problems and express solutions in a way that could be carried out by a computer (Kong, Abelson, & Lai, 2019, Shute, Sun, & Asbell-Clark, 2017; Wing, 2010; Yadav et al., 2017). Although the current study examines a group of students learning about CT through a series of block-based programming activities, it is important to acknowledge that CT has also been touted as a more fundamental skill set that can be applied in multiple and varied subject areas as well as contexts beyond computing (Grover & Pea, 2013).

Purpose

The current study was part of a project designed to support underrepresented middle school students' engagement in activities that incorporated the use of knowledge, skills, and practices represented in the ICT workplace to develop digital artifacts (Cohen, Renken, & Calandra, 2017). Embedded within the larger project, the purpose of the current study was to consider how performance on a multiple-choice computer science assessment called the *CT Quiz* (Ayer, et. al. 2018) was related to the completion of a series of self-paced, mobile app development activities during the one-year after-school computing program. The current study was guided by the following research question: Can student engagement in a series of block-based coding activities during an afterschool program influence their performance on the CT Quiz?

Method

Participants

Study participants were 32 middle school students from 9 schools in an inner-city school district on the southeastern United States who were participating in a free, after-school program that served roughly 2600 students at the time the study took place. While 174 students participated in the afterschool program, assented to participate in research, and provided data at some point in the program, only 32 of these participants provided the researchers with complete data sets related to the current research question and purpose of this study. This was due to the flexible, voluntary nature of the afterschool program. The 32 middle school students who agreed to participate and who produced usable data sets included 16 girls and 16 boys. According to self-reports of race, 18 participants were African American; 5 reported having multiple racial/ethnic backgrounds; and 8 preferred not to self-report race. Participants' ages ranged from 11 to 16, with a mean age of 12.5.

Intervention

The intervention involved participants working on computer science activities in media centers, computer labs, classrooms, and online at 9 middle school sites in an inner-city school district in the southeastern United States. The intervention took place after school for up to 90 minutes twice a week for around 16 weeks. This duration varied somewhat at each school site as each individual school had some control over program implementation. During this time, participants were presented with a series of computer science activities. The instructional design for the activities included self-paced, direct instruction (Kirschner, Sweller, & Clark, 2006), followed by increasingly less structured computational problem-solving activities (Guzdial, 2015). More specifically, the sequence consisted of: a) A guided series of programming steps that lead to the creation of pre-designed mobile apps called *cookbooks*; b) more lightly guided problem-based tasks that allowed participants to continue working on, tweaking, troubleshooting, or remixing existing mobile apps called DIYs; and c) opportunities for participants to work on their own ideas for apps alone or in self-selected teams once they became comfortable with the App Inventor interface. The authors chose MIT's App Inventor as the platform for helping novices learn to code because it is a user-friendly solution that would allow young students to create content for platforms that they commonly use, such as mobile devices (Goode, 2008; Patten, Tissenbaum, & Harunani, 2019). The authors felt that block-based coding was an efficient gateway into coding while also allowing students to showcase varied and rich creative expression (Kafai & Burke, 2017). Both of these affordances of App Inventor were in keeping with our broader goal of increasing access to engaging CS learning experiences, especially for minority students and girls.

The app building activities were created to include computational thinking (CT) concepts and practices as outlined by Brennan and Resnick's (2012) framework. The CT practices embedded in the intervention included abstracting and modularizing, reusing and remixing others' projects, and being incremental and iterative. The CT concepts and related skills included were the use of sequences, loops, parallelism, events, conditionals, and operators. As much as possible, each activity was also designed to connect the app building experiences with relevant, and at times socially responsible themes. Themes for app activities included but not exclusively musical artists, civil rights, games, quizzes, and public health. While activities were presented to participants based on gradual increases of difficulty, the authors would like to note that students were permitted to choose activities out of the suggested sequence, although very few did so. Students who worked out of sequence did so mostly based on their interest in the content/theme of a given activity. Activities could be downloaded and submitted online in a custom-built content management system. During their activity time, participants interacted with school teachers, with graduate student program facilitators, and with undergraduate mentors who were Computer Science majors from 3 local, Historically Black Colleges and Universities.

CT Quiz

While evaluation of the larger project involved multiple and mixed data sources, this paper focuses on the number of apps each participant completed as it related to CT quiz scores. The CT Quiz is an instrument that falls into the classification of a *CT summative instrument* (Roman-Gonzales, Moreno-Leon, & Robles, 2019). In other words, it was designed for a post-test evaluation of student learning after their exposure to a CT-infused curriculum involving mobile app development. The CT Quiz was developed by a team that included experts in CS education, instructional design, and psychometrics, and it was based closely on Yadav et al.'s (2017) definition of CT combined with Brennan and Resnick's (2012) framework for computational thinking concepts and practices (Ayer, et al., 2018). Brennan and Resnick (2012) was used specifically to expand upon the broader definition in order to further identify categories of related, but more concrete CT concepts and practices that are specific to teaching and learning programming (K12CS, 2016). See Table 1 below for the broader CT definition matched to constructs from the framework.

Table 1
Operational Definitions, Practices, and Concepts

Definition	Practices	Coding Concepts (can be embedded within multiple practices)
Breaking down complex problems into manageable smaller chunks of problem	Abstracting and modularizing	*Variables* *Lists* *Sequences* *Loops* *Parallelism* *Events* *Conditionals* *Operators*
Using algorithms to solve problems,	Being incremental and iterative	
Transferring the solution to similar problems	Reusing and remixing other's project	
Determining if an intelligent agent can effectively carry out the solution	Debugging and testing the existing project	

Note: The definitions, concepts, and practices were drawn directly from Yadav et al. (2016) and Brennan and Resnick (2012)

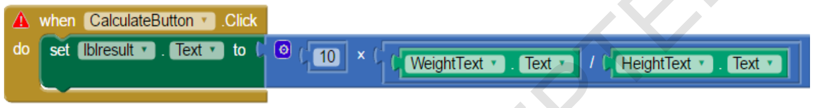
Items in the CT Quiz were designed to allow students to demonstrate CT concepts and to some extent practices by answering multiple choice questions (Ayer, Cohen, & Calandra, 2018). Answer choices were included that were correct, plausible, and incorrect. Plausible in this case did not mean "more correct" than incorrect answer choices. It only meant that these were designed to be more plausible distractors. See Table 2 for a list of items included in the Quiz mapped to CT concepts and CT practices.

Table 2
Quiz items, Practices, and Concepts

CT Quiz Item	*CT Practice*	*CT Concept*

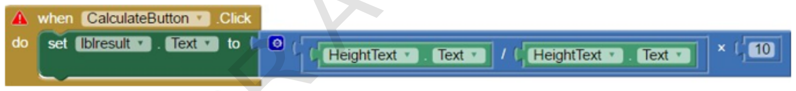
1	Abstraction	Loop
2	Abstraction	Sequence
3	Abstraction	Variable
4	Abstraction	Variable
5	Abstraction, Reuse	Sequence/Event
6	Abstraction, Reuse	Sequence/Event
7	Abstraction	Parallelism
8	Abstraction, Debugging	Operation
9	Abstraction, Reuse	Sequence/Event
10	Abstraction	Loop
11	Abstraction, Debugging	Conditional
12	Abstraction	Conditional

See Figure 1 below for a sample Quiz question. The question in Figure 1 addresses participants' knowledge of how individual pieces of code work together to control objects in the app (abstraction). In addition, students are required to identify and fix an error in a block of code in order to answer the question correctly (debugging).

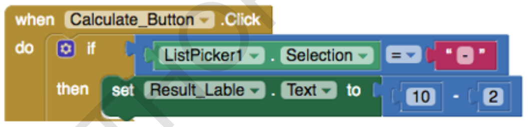


```
when CalculateButton .Click
do  set lblresult . Text to  10 × ( WeightText . Text / HeightText . Text )
```

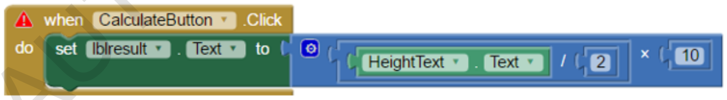
Question: You realize that your code for your app above is doing the calculation wrong for your fitness level. It should multiply the height by two and then divide by 10. Please choose the appropriate code block below to solve the problem.

○ 

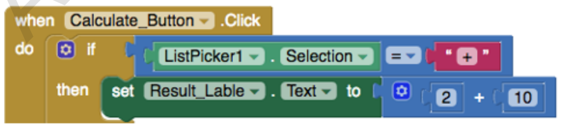
```
when CalculateButton .Click
do  set lblresult . Text to ( HeightText . Text / HeightText . Text ) × 10
```

○ 

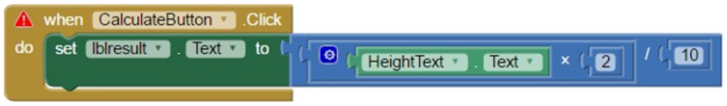
```
when Calculate_Button .Click
do  if   ListPicker1 . Selection = "-"
    then set Result_Lable . Text to 10 - 2
```

○ 

```
when CalculateButton .Click
do  set lblresult . Text to ( HeightText . Text / 2 ) × 10
```

○ 

```
when Calculate_Button .Click
do  if   ListPicker1 . Selection = "+"
    then set Result_Lable . Text to 2 + 10
```

○ 

```
when CalculateButton .Click
do  set lblresult . Text to ( HeightText . Text × 2 ) / 10
```

Figure 1. Sample question (quiz question number 8) from the computational concepts and practices Quiz.

By relevant activities, the authors mean cookbook activities that contained explicit instruction on a given concept as well as exposure to given practices. DIYs and student driven/designed apps are not included in this table because they did not include explicit instruction, however, students who completed these, completed more activities in general, and thus had more opportunity for exposure to given concepts and practices, including some not explicitly included in the cookbooks such as reuse and debugging. See Table 3.

Table 3
Quiz items and Relevant Activities

Quiz Items	*Relevant Activities*	*Number of RA*
1	6, 9, 10	3
2	2, 8, 9, 10, 11	5
3	8, 9, 10, 11	4
4	8, 10, 11	3
5	2, 8, 9, 10, 11	5
6	2, 8, 9, 10, 11	5
7	2, 6, 8, 9, 10, 11	6
8	6, 8, 9, 10, 11	5
9	2	1
10	8, 10, 11	3
11	6, 9, 10	3
12	6, 9, 10	3

Data Collection

Pretesting opportunities were provided for participants at the beginning of each of two school semesters and post testing at the end. Participation in data collection was also voluntary. Researchers were on site to assist participants in completing the CT Quiz in a school computer lab during afterschool time at each school site at the beginning and end of each semester (fall and spring). Thirty-two participants completed the CT Quiz prior to working on the app-building curriculum and after completing each school semester. Due to differences in patterns of attendance, 7 participants completed the CT quiz 3 times rather than 2. For this reason, the authors defined an individual's pretest as the first time a student took the CT Quiz and the posttest as the last time they took it. Participants were instructed to turn in their completed apps by the end of the program. The authors used number of apps completed as a measure of student exposure to the curriculum. App completion in this case meant the participant submitted an at least partially functional mobile application to the online system.

Data Analysis

Quantitative data analysis was conducted by graduate research assistants and a faculty expert in psychometrics. All data was downloaded, stored, de-identified and cleaned in spreadsheets. Analysis was then conducted using the SPSS statistical analysis software. In addition, all apps were downloaded, stored and de-identified after which time they were reviewed for content and creativity as well as instances of CT concepts being implemented. This analysis was guided by the *App Inventor Project Rubric - Computational Thinking through Mobile Computing* created by Sherman et al. (2014). This review was conducted by two doctoral-level graduate assistants: A former Computer Science teacher, and a student of Instructional Design and Technology.

Results

The number of apps participants completed during their exposure to the afterschool curriculum ranged from 1 to 11. The data was normally distributed with a mean app completion rate of 4.6 apps. Pretest scores were normally distributed and ranged from 0% correct to 50% correct, with a mean of 22% correct. As expected, a correlation analysis indicated that the pretest scores did not differ as a function of apps completed (Pearson's $r = .106$, $p = .564$ and Spearman's *rho* = .134, $p = .465$). Posttest scores were moderately skewed to the left and ranged from 0% correct to 83% correct, with a mean of 30.5% correct. The authors calculated a CT Quiz Change score as the individual's posttest score minus their pretest score. The change scores were normally distributed and ranged from -25% to 50%. Both posttest scores and change scores were significantly correlated with the number of apps a participant completed (Posttest: Pearson's *r* = .545, *p* = .001 and Spearman's *rho* = .547, *p* = .001; Change score: Pearson's *r* = .548, *p* = .001 and Spearman's *rho* = .436, *p* = .013). Figure 3 demonstrates the linear relationship between the number of apps completed and the change in CT score from pre- to posttest.

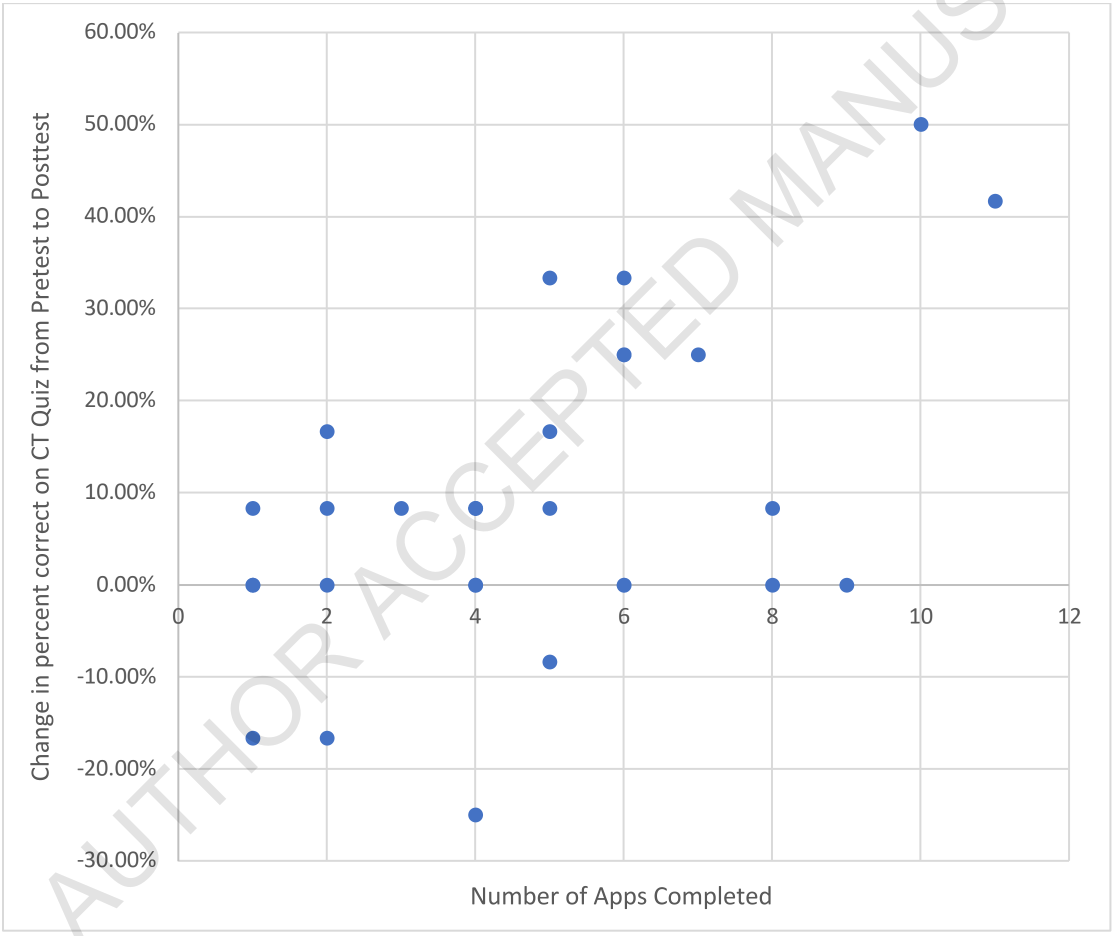


Figure 2. Relation between Change in CT Score from Pre- to Posttest and Number of Apps Completed.

To further explore the positive correlation between change scores on the CT Quiz and the number of apps a participant completed, and thus to more thoroughly answer our research question, the authors determined quartiles based upon the number of apps they completed. See Table 2. In the lowest quartile, participants (*n* = 9) completed 1 or 2 apps. In the second quartile, participants (*n* = 8) completed 3 or 4 apps. In the third, participants (*n* =9) completed 5 or 6 apps; and in the fourth, participants (*n* = 6) completed between 7 and 11.

Table 3
Number of apps completed and number of participants in each app completion quartile

Quartile	Range of apps completed	Number of participants (n)
1	1-2 apps	9
2	3-4 apps	8
3	5-6 apps	9
4	7-11 apps	6

The authors were interested in the percentage of participants in each quartile with gains in their score (i.e., change score > 0), a loss in their score (i.e., change score < 0), or no change (i.e., change score = 0). The percentage of participants who scored higher at posttest than at pretest jumped from 33.3% in Quartile 1 (Q1) to 66.7% in Quartile 4 (Q4). See Figure 3 for mean change in CT scores across groups.

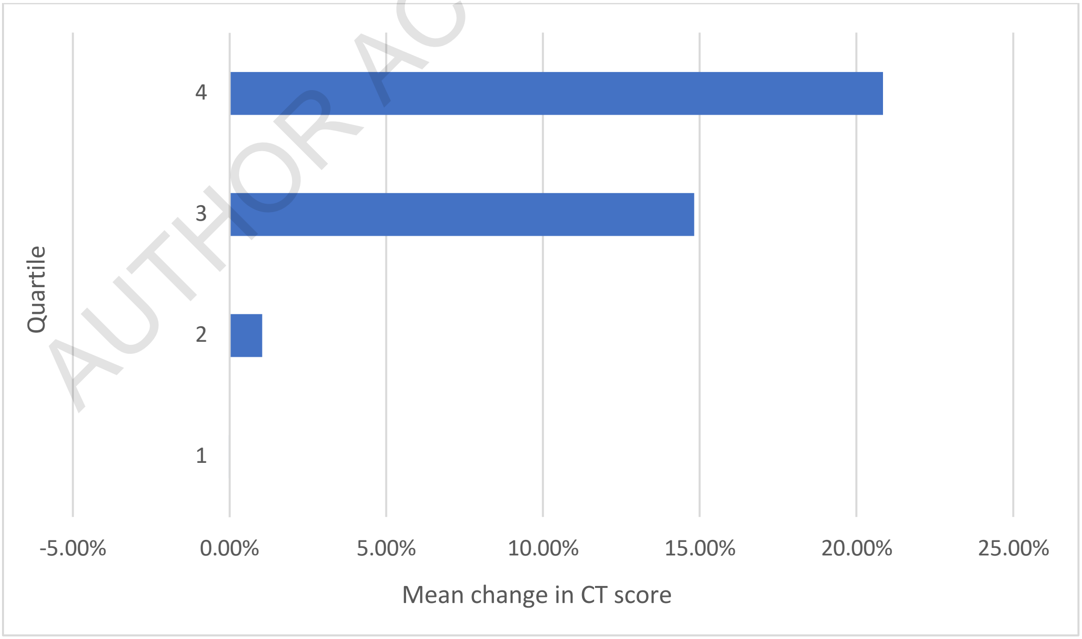


Figure 3. Mean change in percent accurate on the CT Quiz from pretest to posttest within each quartile.

Further, there were 4 participants whose percentage of correct responses decreased from pre to posttest. These participants were only in quartiles 1 – 3; in other words, none of the participants who completed more than 6 apps performed more poorly at posttest than they did at pretest. One way to think about performance on the posttest is with regard to performing at chance. With 4 multiple choice answers from which to select the correct answer, performing at chance would be equivalent to 25% accuracy. For the top quartiles, 83.33% of the participants in Q4 and 77.78% of the participants in Q3 performed above chance on the posttest. In contrast, 0%, and 22.22% performed above chance in the other quartiles, Q2 and Q1, respectively. Of the two students in Q1 that scored above chance, the first student's grade stayed the same from the fall post-test to the spring post-test while the other's increased from the fall (16.67%) to the spring (33.33%) having had the chance to work on 2 apps.

The authors conducted a One-way Analysis of Variance (ANOVA) with CT change score as the dependent variable and quartile as the factor. A one-way ANOVA was selected because it is a statistical test designed to assess whether or not there are statistically significant differences in means across more than two independent groups. In this case, the authors divided participants into independent groups as a function of the number of apps they completed and the authors were interested in how this group membership related to the average CT change scores in each group. Specifically, the authors were interested in whether or not there were differences in CT change scores across the groups. CT change scores were significantly related to number of apps completed in the 4 quartiles (*F(3, 28) = 3.616, p = .025*). To determine statistically significant differences among the 4 groups, the authors ran post hoc analyses. A Tukey HSD test, LSD test, and a Bonferroni test each showed no statistically significant difference between change score for participants in Q1 versus Q2 (Tukey: *p* = .999, LSD: *p* = .886, Bonferroni: *p* = 1.000). Participants in Q1 versus Q4 came closer to showing a significant difference with a Tukey HSD test and Bonferroni test (Tukey: p = .056 and Bonferroni: p = .074). The LSD test, however, suggested a significant difference between Q4 and Q1 and Q2 but not Q3 (Q1: *p* = .012, Q2: *p* = .019, Q3: *p* = .446).

The authors must point out that, none of the students worked on apps using App Inventor immediately prior to the pretest, although it was not possible to confirm whether or not any of the schools were working on computational thinking related activities during the school day. The authors did, however, ask students about their related prior knowledge and prior technology use, and there was no noticeable relationship between reported prior knowledge and pretest scores. Most importantly, participants' pretest data was collected at the beginning of their participation in the program, either at the start of the fall or start of the spring semester. This meant that they were not provided with any app building experience through the program in question prior to the pretest. That being said, 7 of the participants took the test 3 times rather than 2 times. They completed 2, 1, 4, 5, 10, 7, and 11 apps, respectively. These participants were spread across app completion quartiles: Quartiles 1 (n = 2), Quartile 2 (n=1), Quartile 3 (n=1), and Quartile 4 (n = 3). All but 2 of the 7 students mentioned above showed positive gains in CT quiz scores of time but two who were in Quartiles 1 and 2.

Artifact analysis data were used to help the researchers further explore the quantitative findings. First, a review using Sherman et al.'s (2014) rubric showed that students in the 4th quartile and to some extent also the 3rd not only completed more app activities, but they progressed through more activities of increasing difficulty, which led them to be exposed to more of the CT concepts represented in the CT Quiz. See Table 4.

Table 4
Total Number of Apps Completed by Participants in Each Quartile

Quartile	*Number of Participants*	*Total Number of Apps*	*Total Number of Relevant Apps*
1	9	13	5
2	8	31	9
3	9	50	21
4	6	53	24

Artifact analysis further indicated that, as compared to their classmates who completed fewer and more basic apps, students in higher app-completion quartiles were not only completing more apps, but they were also expanding upon their cookbook apps by adding design, media, and functional tweaks to the existing apps, thus potentially having more exposure to CT concepts included in the Quiz. Students in the 4th quartile also enjoyed the opportunity to work on at least one DIY activity, which would indicate their reaching a certain comfort level with the App Inventor interface and CT concepts. Some examples of DIY apps included a fan app for various celebrities, a healthy living advice app based on a users' age, a very colorful adapted version of the space invaders game, and an app designed to "hypnotize" users.

Conclusion

Results demonstrated that in this particular context, with these 32 participants, exposure to computing activities that include CT concepts of gradually increasing complexity using App Inventor seemed to enhance student performance on the CT Quiz. However, there appears to be a threshold for which this held true. In this case, completing 3 or fewer apps resulted in distinguishably different learning outcomes than completing 7 or more over the course of a year-long intervention. In addition, students who completed more apps were also able to explore the app building process in more details via exposure to more DIY activities. Indeed, sustained attendance in afterschool programs and support for completing and progressing through after-school curricula such as the activities presented here is imperative because, as may have been the case in this study, it was evident that exposure to more and a greater variety of activities equaled better performance on a measure of computational thinking.

Although the activities in this study did cover CT concepts that were represented in the CT quiz, future research will need to continue to consider at a finer grain size the overlap in content taught in specific modules that participants complete and their performance on an assessment that covers multiple CT concepts. Such an investigation should provide details about the generalizability of skills clearly aligned with one concept to others that are less directly aligned. Our data was collected at the end of each semester, so there was a brief relative delay from task completion to assessment, but future work should also explore the temporal transfer of gains in CT conceptual knowledge. The current findings do not address transfer to a different app building or coding language platform. Future research must focus on generalizing findings like these across such facets of transfer. As recommended by colleague, fifth author, and college (2019), via the creation of the CT Quiz and its early implementation as described in this paper, the authors hope to contribute to a battery of commonly used instruments for assessing commonly understood variables associated with Computational Thinking.

Because this study was conducted in a large, inner-city afterschool program with a group of minority middle school girls and boys who self-selected into the program in question, this study answers a call by Mouza et al. (2016) that, "future research should examine the benefits of CS after-school programs in more diverse contexts, in terms of both gender and ethnicity" (p. 99). Part of the value of this study is that it provides some evidence that with proper scaffolding, gradually increasing levels of challenge, and sustained interest and attendance, after school programs that include CS curricula may lead to cognitive gains related to CT. Although these results are not generalizable due mostly to sample size, this intervention and corresponding measure show enough promise for them to be to be further tested, refined, and validated in multiple and varied contexts, thus making it a worthwhile addition to the larger conversation surrounding broadening participation in computing via informal computer science educational opportunities.

References

Asprey, W. (2016). *Participation in Computing: The National Science Foundation's Expansionary Programs*. Switzerland: Springer.

Ayer, T., Cohen, C., & Calandra, B. (2018). *Computational Thinking Assessment.* Paper presented at the American Educational Research Association (AERA) 2018 conference, New York, NY.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*, 1–25. Retrieved from

Code.org (2019, January). Promote computer science. Retrieved from https://code.org/promote

Cohen, J., Renken, M. & Calandra, B. (2017). 21st century skills and the ICT workplace: Do urban middle school students feel prepared? *Tech Trends. 61*(4), 380-385.

Cuny, J. (2016, February). CS Education: Catching the Wave. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 3-3). ACM.

Equity in Computer Science Education. (2019). Retrieved from https://k12cs.org/equity-in-computer-science-education/

Goode, J. (2008). Increasing Diversity in K-12 computer science: Strategies from the field. In *ACM SIGCSE Bulletin* (Vol. 40, pp. 362–366). ACM.

Goode, J. (2010). Connecting k-16 curriculum & policy: making computer science engaging, accessible, and hospitable for underrepresented students. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 22–26). ACM.

Google Inc. & Gallup Inc. (2016). Diversity Gaps in Computer Science: Exploring the Underrepresentation of Girls, Blacks and Hispanics. Retrieved from: http://goo.gl/PG34aH

Grover, Suchi. (2017). Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In Rich, Peter J. & Hodges, Charles B. (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 269–288). Cham, CH: Springer International Publishing.

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12 (pp. 57–62). ACM Press. Retrieved from https://doi.org/10.1145/2591708.2591713

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.

Guzdial, M. (2015). What's the Best Way to Teach Computer Science to Beginners? *Communications of the ACM*, *58*(2), 12-13.

Guzdial, M., Ericson, B., Mcklin, T., & Engelman, S. (2014). Georgia computes! An intervention in a US state, with formal and informal education in a policy context. *ACM Transactions on Computing Education (TOCE)*, *14*(2), 13.

Howard, A. (2016). Why fixing tech's gender and racial gaps is more crucial than ever. *Tech Republic.* Retrieved from https://www.techrepublic.com/article/why-fixing-techs-gender-and-racial-gaps-is-more-crucial-than-ever/

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

K–12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org

Kafai, Y. & Burke, Q. (2014). *Connected Code: Why Children Need to Learn Programming*. Cambridge, MA: MIT Press.

Kafai, Y. B., & Burke, Q. (2017). Computational Participation: Teaching Kids to Create and Connect Through Code. In P. J. Rich & C. B. Hodges (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 393–405).

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, *41*(2), 75–86.

Kong, S., Abelson, H., & Lai, M. (2019). Introduction to Computational Thinking Education. In, S. Kong & H. Abelson (Eds.), *Computational Thinking Education* (pp. 1-10). Springer Open.

Lachney, M. (2018). Computational communities: African-American cultural capital in computer science education. *Computer Science Education*, 1–22.

Lee, W. (2015) How tech's lack of diversity leads to racist software. *SFGate.* Retrieved from http://www.sfgate.com/business/article/How-tech-s-lack-of-diversity-leads-to-racist-6398224.php

Margulieux, L., Ketenci, T. A., & Decker, A. (2019). Review of measurements used in computing education research and suggestions for increasing standardization. *Computer Science Education*, 1–30.

Patton, E.W., Tissenbaum, M., & Harunani, F. (2019). MIT App Inventor: Objectives, Design, and Development. In, S. Kong & H. Abelson (Eds.), *Computational Thinking Education* (pp. 31-49). Springer Open.

Rankin, Y., & Thomas, J. (2017). *Moving Students of Color from Consumers to Producers of Technology.* Hershey, PA: IGI Global.

Roman-Gonzales, M., Moreno-Leon, J., & Robles, G. (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In, S. Kong & H. Abelson (Eds.), *Computational Thinking Education* (pp. 79-98). Springer Open.

Ryoo, J., Goode, J., & Margolis, J. (2015). It takes a village: supporting inquiry- and equity-oriented computer science pedagogy through a professional learning community. *Computer Science Education*, *25*(4), 351–370.

Saran, C. (2017). Lack of diversity risks creating big gaps in AI. *Computer Weekly.* Retrieved from http://www.computerweekly.com/news/450411202/Lack-of-diversity-risks-creating-big-gaps-in-AI

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158.

The White House. (2016, January 30). *Computer science for all*. Retrieved from: https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Wing, J. M. (2010). Computational thinking: What and why? Unpublished manuscript. Pittsburgh, PA: Computer Science Department, Carnegie Mellon University. Retrieved from https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). 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.