Summer Programming Camps – Exploring Project-Based Informal CS Education in a Rural Community

Carla DE LIRA[1]
Rachel WONG[2]
Olufunso OJE[1]
Gabriel NKETAH[1]
Olusola ADESOPE[1]
Alireza GHODS[1]

[1] Washington State University, Pullman, WA, United States of America
[2] University of Tennessee, Knoxville, TN, United States of America

DOI: 10.21585/ijcses.v5i4.145

Abstract

Current research has not fully explored how summer programming camps can help students increase motivation and interest to pursue computing career, and their programming knowledge. Informal CS education through summer programming camps provides K-12 students the opportunity to learn how to code through fun and interactive activities outside of their typical classroom experiences. In this study, we examined the effectiveness of a weeklong summer programming camp for promoting students' motivation and interest in programming, and their programming knowledge. Participants were 19 middle school students from rural Washington. Students participated in a project-based learning approach through game development in Python. Using a within-subjects design, we analyzed students' pre and post motivation and knowledge assessment scores. Results from the analysis indicated a significant improvement in post-test programming knowledge scores (d = 0.93). The findings also indicated that students were able to achieve basic abstraction and algorithmic thinking but not code analysis and debugging skills. On their motivation to pursue computing careers, the results did not show any difference before and after the camp due to their prior existing interest in attending the camp.

Keywords: Computer science education, pre-college programs, STEM, programming camps, K-12 education

1. Introduction

The number of individuals graduating with a Science, Technology, Engineering, and Mathematics (STEM) major remains low despite the increase in STEM jobs in the United States (Bureau of Labor Statistics, 2019; National Science Board, 2016; Xianglei & Weko, 2009). Two plausible reasons include the lack of interest in pursuing STEM-related courses, and the lack of early opportunities and exposure to STEM (Tai et al., 2016.). This issue is further exacerbated in computing where there is rapid demand for talent in the tech industry in the United States, but not enough of graduates in computing-related degrees (Zweben & Bizot, 2020). According to the Bureau of Labor Statistics, computing occupations, such as software developers and computer programmers, are projected to grow 13% between 2020 and 2030 (Bureau of Labor Statistics, 2019). It is then important to ensure that we spark interest in computing early among K-12 students in hopes that they become the next generation to maintain and develop our technological infrastructures. Early exposure to computing opportunities, especially for girls, is important as it may increase a child's interest in computing, improve their perceptions, and eliminate gender stereotypes (Bagiati et al., 2010; Tai et al., 2016) In fact, it has been shown that early exposure to computing prior to high school yields a higher chance that their interest in computing maintains into higher education (Christensen et al., 2014; Hirsch et al., 2017; Taub et al., 2012) Out-of-school activities or informal learning experiences through STEM camps is one potential way to provide early exposure to STEM (Bell et al., 2009, p. 20; Cabrera et al.,

2021; Mohr-Schroeder et al., 2014), especially computing (DeWitt et al., 2017b; Frye et al., 2016; Master et al., 2017).

Informal learning environments go beyond the traditional classroom and provide a casual learning experience for students (Roberts et al., 2018). Within computer science (CS) education, these learning opportunities commonly introduce computing concepts in a hands-on approach or relatable manner that may be beneficial for supporting formal computer science (CS) education in the future (DeWitt et al., 2017a; Franklin et al., 2013; Lakanen & Kärkkäinen, 2019; Xianglei & Weko, 2009). Currently, many school districts in the United States still do not incorporate programming as part of their STEM curriculum, due to the lack of resources, such as finding teachers who can teach it (Warner et al., 2019). Informal CS learning opportunities may be the only time students in a particular region would be able to engage in programming outside of the classroom and possibly prior to college (Warner et al., 2019). Informal STEM learning opportunities, such as programming camps, are often offered during the summer after the school year (Frye et al., 2016; Roberts et al., 2018; Webb & Rosson, 2011). Since knowledge loss typically occurs over summer breaks due to the lack of access to learning opportunities (McCombs et al., 2011), free informal STEM opportunities, like programming camps, are particularly important for students from low socioeconomic backgrounds who otherwise may not have access (Lusa Krug et al., 2021). Since programming camps can use various STEM concepts as a context for learning how to code (LePendu et al., 2020; Nite et al., 2020), these programs can provide the opportunity to engage in STEM topics covered during the school year while also introducing coding concepts.

Given the positive effects of summer programming camps on students' interest in computing, the research team developed a free summer camp for middle school students in rural Eastern Washington. A one-time programming camp was previously offered in the region, albeit only for middle-school-aged girls. Due to the lack of programming resources in the area, this camp was designed for middle school students. The study has two broad aims.

First, we are interested in examining the impact of participation in a week-long summer programming camp on students' motivation in programming and interest in pursuing a programming-related career. Research suggests that even a short week-long exposure to STEM activities may increase students' interest in STEM and positively influence their perceptions about STEM (National Science Board, 2016; Xianglei & Weko, 2009). We have also seen this reflected in longitudinal studies. Girls who were exposed to computing at a programming camp maintained an interest in programming over time (Outlay et al., 2017).

We are also interested in examining whether participation in the week-long camp is sufficient exposure to increase students' knowledge of programming and their ability to apply programming concepts. Franklin et al.'s study found that exposing students to two weeks of programming was sufficient for imparting computer science knowledge (Franklin et al., 2013). Programming provides the opportunity to exercise several computational thinking skills, such as understanding abstraction, problem formulation, and debugging for K-12 students (Lye & Koh, 2014). Thus, we are also interested in assessing students' computational thinking (CT) skills based on their programming knowledge performance. Despite the little research on learning to code through informal learning environments (i.e., programming camps), preliminary research indicates that informal learning experiences are effective in teaching code to students (Akcaoglu, 2014; Denner et al., 2012; Wang & Frye, 2019; Zamin et al., 2018). It is less clear as to how informal learning experiences in computing are effective in teaching computational thinking skills, especially since there is still ongoing discussion among scholars as to what CT comprises and ongoing efforts to measure CT skills (Shute et al., 2017; Werner et al., 2012)

Second, we are interested in the effectiveness of a hands-on project-based approach in helping students learn and retain programming concepts. In this approach, key concepts are interwoven into each step of the project that students are required to work on. Essentially, students learn and apply those key concepts simultaneously. To address these two broad aims, this study seeks to answer the following research questions:

> **RQ 1)** How does participation in project-based learning influence students' motivation and interest before and after a short informal programming camp?
>
> **RQ 2)** How does project-based learning influence students' programming knowledge before and after a short informal programming camp?

2. Related Work

Informal learning experiences are frequently offered outside of the classroom and structured curricula (Franklin et al., 2013; Roberts et al., 2018). Examples of informal learning environments include after-school programs,

museum/field trips, and summer camps (Hofstein & Rosenfeld, 1996). In such environments, instructors and organizers are typically concerned with engaged participation, affective outcomes, and developing interest among students with loose learning objectives set for the duration of the informal learning opportunity(Hofstein & Rosenfeld, 1996; Stewart & Jordan, 2017).

Such opportunities are valuable for a couple of reasons. The emphasis on engaging participants and developing interest is especially important for female students who tend to lose interest in STEM while in middle school and through post-graduate education (Bagiati et al., 2010; Master et al., 2017). In addition, without sufficient exposure to STEM opportunities, students may develop a negative attitude towards STEM (Weinberg et al., 2011). Existing studies provide insights on the positive impact informal STEM opportunities have on students in future college major choices and interest in a STEM-related field (Miller et al., 2018; Weinberg et al., 2011).

In K-12 computer science education, there has been a gradual increase in recent years in summer programming camps as a popular form of an informal learning opportunity to stimulate interest in pursuing computer science(Bell et al., 2009; Bureau of Labor Statistics, 2019). These programming camps offer students the opportunity to delve into computing concepts that are largely not covered in many K-12 school curricula in the nation, especially in elementary and middle schools (Fields et al., 2015; Frye et al., 2016). The likelihood of a K-12 school curriculum that covers computing concepts becomes less in rural communities (Code Advocacy Coalition, 2018). For students in these underserved areas, a programming camp provides a learning opportunity in STEM that may be fun and engaging through an informal learning environment (Roberts et al., 2018).

2.1 Structure of Programming Camps for Middle School Students

One of the aims of programming camps is to provide students with an opportunity where they can learn problem-solving skills, have fun with programming tasks, and interact with their peers with similar interests (Adams, 2010). These camps cater to a range of students from elementary school (Chaudhary et al., 2016) to high school (Al-Bow et al., 2009). However, there has been a focus to provide programming opportunities particularly to middle school students (DeWitt et al., 2017b). Choices made in middle school can impact future education and career pursuits (Al-Bow et al., 2009; Wang et al., 2019). A major predictor of a student pursuing a STEM career upon graduating high school is their interest at the start of high school (Lakanen & Kärkkäinen, 2019; McCombs et al., 2011). Since interest in STEM careers may decline as a student matures (Ayar & Yalvac, 2016), it is crucial to spark interest in STEM in middle school students before they start high school (Hofstein & Rosenfeld, 1996; Xianglei & Weko, 2009).

Programming camps for middle school students are often in the form of hands-on workshops that utilize block programming languages, such as Scratch, or text-based languages, such as Python (Bryant et al., 2019). Such programs provide guidance in completing coding activities (Austin & Pinkard, 2008; Bagiati et al., 2010; Bell et al., 2009; Stewart & Jordan, 2017; Wang et al., 2019; Xianglei & Weko, 2009). Such camps have been found to be effective in generating interest in computer science and teaching students of varying backgrounds how to code (Maiorca et al., 2021; Weinberg et al., 2011).

Interestingly, although programming camps generate interest in computer science, little research has been conducted to examine how well these camps promote the acquisition and retention of students' programming knowledge. More specifically, there is a lack of research on the effective teaching methods in these informal learning environments. Thus, this study seeks to explore whether a project-based programming camp is able to foster learning of challenging programming concepts.

2.2 Project-based Programming Camps

In K-12 computer science education, there have been some efforts to discuss how to support students' growth in programming knowledge through project-based learning in informal learning environments, such as programming camps (Fields et al., 2015). Project-based learning is one of the most common teaching approaches in introducing K-12 students to STEM fields ("2018 NSSME+," 2018.; Adams, 2010; Austin & Pinkard, 2008; Burack et al., 2018; DeWitt et al., 2017; Jones, 2019). This approach allows students to apply taught concepts to real-world experience through a project (Hugerat, 2016; Webb & Rosson, 2011). Project-based learning differs from traditional learning in that the project plays the main role in the curriculum. Students learn about concepts as they progress in their project, which is often student-driven with some guidance from instructors/organizers. Project-based learning in STEM is also an effective way to promote K-12 students' STEM career interest (Al-Bow et al., 2009; DeWitt et al., 2017b). However, the effectiveness of project-based learning in gaining skills to prosper in STEM is largely unexplored within informal learning environments. As concepts in a project-based learning

approach are introduced as students need them, it is unclear whether such concepts are retained at the end of their learning experience.

2.3 Project-based Programming Camps for Increasing Motivation & Interest

Project-based programming camps are typically organized to provide programming knowledge for middle school students to start working on their projects by the first or second day. Webb and Rosson held a week-long programming camp for middle school girls using Alice, a visual block programming environment, to gradually introduce programming concepts that they would need to create their individual 3D story (Webb & Rosson, 2011). At the end of the camp, they found that students were more interested in pursuing computer programing. In a shorter two-day programming camp, this method of gradually introducing just enough programming concepts to middle school students was also effective in promoting interest in computing careers (Outlay et al., 2017).

Another characteristic of project-based programming camps is the ability for students to share their completed projects at the end of the camp to instructors, friends, and even family(Bryant et al., 2019). In other camps, students have also created research posters to showcase their projects (Wang et al., 2019). Incorporating a project presentation component in a project-based programming camp might enhance students' sense of accomplishment by the end of the program (Sadler et al., 2018; Weinberg et al., 2011).

In general, project-based programming camps have been found to be very effective in generating middle school students' interest and motivation in computing careers. By providing as- needed information and concepts so students can complete their projects helps to build their confidence from the very beginning. The presentation component also allows them to share their success with others (Adams, 2010; National Science Board, 2016; Stewart & Jordan, 2017) .

2.4 Project-based Programming Camps for Increasing Programming Knowledge

The desired outcomes for middle school students attending programming camps are an increased interest in programming careers, increased programming knowledge, and enjoyment in completing programming activities. Research highlights three different ways instructors can assess participants' knowledge. Ericson and McKlin utilized a 10-item multiple choice pre and post survey to assess middle and high school participants' programming knowledge. Results showed significant increases from pretest to posttest across different programming concepts, such as loops, variables, conditional statements etc., (Ericson & McKlin, 2012). In another study, students were asked to rate how much they knew about programming on a scale of 0 (nothing) to 5 (expert) after the camp. Seventy-three percent reported an increase in programming knowledge while 27% reported no change (Mohr-Schroeder et al., 2014). Unlike Ericson and McKlin, Franklin et al. analyzed participants' programming projects on Scratch to assess whether students acquired programming concepts (Franklin et al., 2013). This assessment allowed researchers to conclude that at the end of their two-weeklong camp, students successfully mastered event-driven programming, message passing, state initialization, and say/sound synchronization (Franklin et al., 2013). Interestingly, less attention has been paid to the assessment of more foundational type concepts such as variables, loops, conditional statements, data structures, and functions.

3. Method

In the present study, we examined the impact of a one-week project-based informal computer programming summer experience on students' perceptions of programming and programming knowledge in rural Washington where the availability of such opportunities is sparse.

3.1 Sample Information and Research Design

Nineteen middle school students (M_{age} = 12.72; SD_{age} = 0.96; Girls = 13, Boys = 6) participated in the summer programming camp. Majority of the students identified as Asian (n = 10), followed by Caucasian (n = 3), and Black (n = 1). The other students either preferred not to answer (n = 4) or indicated that their race/ethnicity was not listed (n = 1). Students self-selected based on their interest (or parents' interest) to attend the week-long summer programming camp at a large pacific northwestern university. Students who were a part of this programming camp had some familiarity with programming concepts before the week-long program. To examine the effect of the programming camp exposure on students' programming motivation and knowledge, we used as a within-subjects research design. Students completed pre- and post- motivation surveys and learning assessments. This study was deemed exempt by the University's Institutional Review Board.

Table 1. Daily Breakdown of Programming Camp

Day	Programming Concept	Activities
1	A quick introduction to Python, run code from the command line, introduction to variables, lists, and Turtle library	Make snake game screen using Turtle library to make simple shapes.
2	Introduction to loops, conditional statements, user input, generating randomness	Implement functionality for snakehead placement and apple placement.
3	Introduction to functions	Implement functions and further snake game improvements.
4	No new content coverage	Finish snake game.
5	No new content coverage	Have fun and help students make improvements to the snake game.

3.2 Data Collection Tools

To examine the influence of the programming camp on students' motivation and interest in programming, we administered a 20-item survey. The 20-item survey was adapted from exiting measures on students' interest and motivation in STEM (see Glynn et al., 2011; Korkmaz, 2017; Yadav et al., 2011), in addition to a handful of researcher-developed questions. The items were further divided into 5 subscales, Career Interest, Interest, Value, Critical Thinking, Proficiency. Students were asked to indicate the extent to which they agreed or disagreed with each of the following items using a 5-point Likert scale, where 1 = Strongly Disagree and 5 = Strongly Agree. See Table 2 for the list of items that were included on the survey and for each subscale.

To assess programming knowledge, we administered the same 10-item multi-step programming knowledge assessment before and after the camp (see Appendix A). The 10 items used in this programming knowledge test were researcher-developed. The items were developed around the concepts that were taught in the programming camp. The 10 items on the pre- and post-programming knowledge assessment were categorized into 3 broad categories related to CT skills: basic abstraction/operations, code analysis, and code writing. We decided to center on these CT skills, since we are able to connect the programming questions which exercise abstraction (basic abstraction/operations), problem formulation (code writing), and debugging/analysis (code analysis). However, we recognize that there is still ongoing discussion on what constitutes as CT skills within literature (Shute et al., 2017).

The basic abstraction/operations category contained questions related to basic programming operations like printing out values, mathematical operations, string concatenation, variables etc. In the code analysis category, snippets of code were provided to the participants for analysis. They were required to write out the output of the code snippet. This category also tested if participants could detect issues with the code snippet. The code snippets focused mainly on loops and conditional statements. In the code writing category, challenges were presented to the participants, and they were required to write code to solve the challenge. For example, the participants were asked to write code that would print a phrase 10 times. The expectation was for the participants to use loops rather than write a print statement 10 times.

Table 2. Survey Items for Pre- & Post- Motivation

Career Interest
19. My knowledge of computer programming will help me choose a career in computing.
20. I am interested in a career in programming.
11. The computer programming skills I learn will be useful in my life.
14. I put effort into learning about computer programming.
12. I believe I can master computer programming knowledge and skills.
10. I will use computer programming skills in the future.
4. I will take more computer programming courses if I have the opportunity.
Interest
13. I enjoy learning about computer programming.
5. I have a special interest in mathematical processes.
1. Learning computer programming is interesting.
Value
2. Having an understanding of computer programming is valuable.
15. Understanding computer programming is important to me.
Critical Thinking
3. The challenge of solving problems using computer programming skills is appealing to me.
8. It is fun to try to solve complex computer programming problems.
9. I am willing to learn challenging computer programming problems.
18. What I already know about computer programming will help me think critically.
Proficiency
17. I am proficient in computer programming.
16. I know how to write computer programs.
7. I can easily understand the relationship between figures.
6. I can better learn instructions with the help of mathematical symbols and concepts.

3.3 Scoring

To calculate the total for each subscale on the motivation survey, we summed the students' responses for the items on each subscale. Both the pre- and post- motivation surveys had high reliability ($\alpha = .94$, and $\alpha = .96$, respectively). The individual subscales for the pre- and post- surveys also had moderate to high reliability with Cronbach's alpha ranging from .60 to .90.

To score the programming knowledge test, we graded the assessments based on an answer key developed by the instructors. We used the overall score of the assessments to determine whether there was a difference before and after the programming camp. First, the team evaluated and grouped questions based on programming concepts and the question type: basic concepts, original code, and code analysis. Second, two members of our team rated the mastery level for students' answers to each programming concept question from 1 (low understanding) to 5 (high understanding). Full agreement in inter-rater reliability was obtained between the two graders. Scores were obtained for the three broad categories basic operations (15 points), code analysis (15 points), and code writing (20 points), and the sum of these three categories provided the overall score (50 points). There was moderate to high reliability for the pre-assessment categories: basic operations, $\alpha = .50$, code analysis, $\alpha = .60$, and code writing, $\alpha = .84$. There was also moderate to high reliability for the post-assessment categories: basic operations, $\alpha = .59$,

code analysis, $\alpha = .46$, and code writing, $\alpha = .87$. Finally, there was moderate to high reliability for the pre- and post- assessment as a whole, $\alpha = .83$ and $\alpha = .74$, respectively. Examples of low and high rated answers are included in Appendix B.

3.4 Procedure

The computer programming camp was held in a spacious computer lab on the campus of a large university in the Pacific Northwest of the United States. Laptops equipped with the appropriate software were provided to students during the duration of the programming camp. On the first day, students completed both the pre-motivation survey and the pre-knowledge assessment test before commencing the camp activities. Each day, the instructors started with an overview of the day's lesson. Lessons on the programming concepts were interwoven into a hands-on project-based activity of building a snake game app from starter code. Instructors started with a brief lecture on core concepts for the day before walking through their own example code as students paid attention. Following this, students were given ample time to apply their newly acquired programming knowledge to the development of their game app. Instructors and teaching assistants provided one-on-one instructional support as needed. Each day comprised of at least two lectures and two sessions of individual coding time to develop the game app. This process is important because it allows students to examine how their knowledge of programming translates directly into the design and functionality of their game, which is likely to increase their appreciation and interest in programming. On the last day of the camp, students completed the post-motivation survey and the post-knowledge assessment test. An outline of each day's programming content coverage as it relates to the activities and project is provided on Table 1.

4. Results

To address our research question, separate analyses were conducted for the motivation and knowledge assessment measures. The results section is organized around these two analyses.

4.1 Motivation Analysis

To address RQ1, we analyzed data from pre-and post- motivation surveys which were administered on the first and last day of the programming camp. Nineteen students completed the pre- and post-motivation survey before and after the camp. There were 2 missing data entries for the pre-motivation survey and 2 missing data entries for the post-motivation survey. As the data was missing at random, we employed the EM algorithm to compute missing data points. The data was normally distributed for each of the motivation subscales. Table 3 provides the descriptive statistics for each individual subscale's score.

Table 3. Descriptive Statistics for Motivation Subscales

	Pre		Post		
Assessment	M	SD	M	SD	Cohen's *d*
Career Interest	26.62	4.80	26.95	4.60	0.16
Interest	12.58	1.68	12.21	1.87	-0.32
Value	8.42	1.35	8.05	1.35	-0.44
Critical Thinking	16.47	2.80	16.00	2.92	-0.39
Proficiency	13.21	2.37	14.53	2.41	0.99

Paired-samples t-tests were conducted to assess students' change in motivation score for the five subscales (career interest, interest, value, critical thinking, proficiency). Results showed that there were significant differences in students' score for the proficiency subscale, $t(18) = 4.29$, $p < .001$. Specifically, students self-reported higher proficiency after the programming camp ($M = 14.53$, $SD = 2.41$) as compared to before the camp ($M = 13.21$, $SD = 2.37$; $d = 0.99$). There were no significant differences for career interest, $t(18) = 0.59$, $p < .57$; interest, $t(18) = -1.38$, $p = .19$; value, $t(18) = -1.93$, $p < .07$; and critical thinking, $t(18) = -1.69$, $p = .11$.

Table 4. Descriptive Statistics for Programming Knowledge Assessments

	Pre		Post		
Assessment	M	SD	M	SD	Cohen's *d*
Overall	21.37	9.76	30.05	8.66	0.93
Basic Operations	10.84	3.67	13.26	2.90	0.71
Code Analysis	4.89	3.74	6.74	3.69	0.50
Code Writing	5.63	4.30	10.05	5.40	0.88

4.2 Programming Knowledge Analysis

To address RQ2, paired-samples t-tests were conducted to analyze the differences between the pre- and post-programming knowledge assessments. Both the overall assessment scores and the scores for each of the three categories were analyzed separately. Each of the knowledge assessment categories were considered (Basic Operations, Code Writing, Code Analysis, and Overall scores). Normality and outlier tests were performed on the overall scores (each category is a sub-score of the overall score). No outliers were detected. The assumption of normality was also not violated, as assessed by Shapiro-Wilk's test ($p = .313$). There were no outliers in the data, as assessed by inspection of a boxplot. Overall results indicate a statistically significant mean increase in programming knowledge, $t(18) = 5.82$, $p < .01$ (See Table 4 for descriptive statistics).

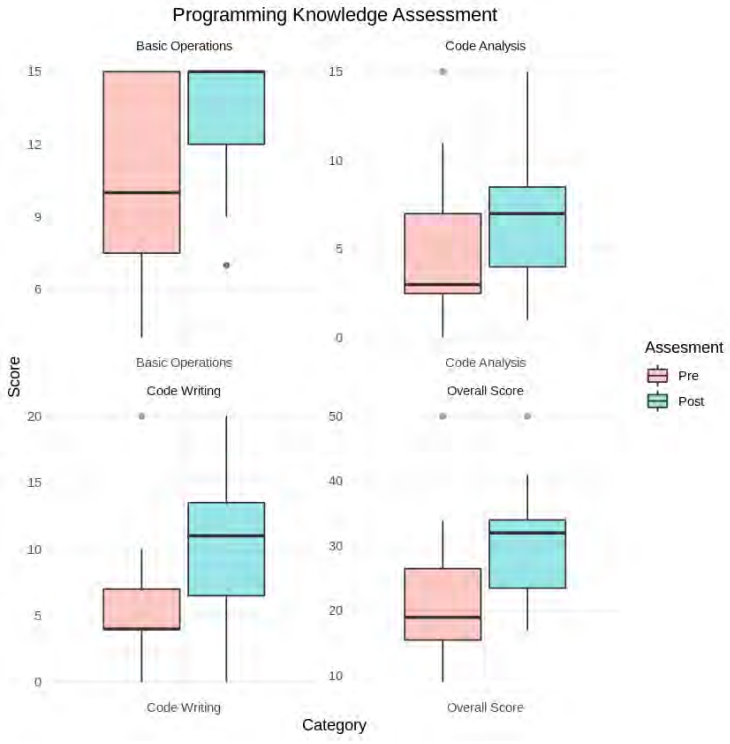

Figure 1. Programming Knowledge Assessment Boxplot

The results show that the students had significantly higher scores from the post-assessment. The mean difference between the pre- and post-assessment score also seem to suggest that the participants gained a lot of programming knowledge from the camp (Figure 1).

The results also show that the participants performed better in Basic Operations and Code Writing sections of the assessment. However, the increase in Code Analysis after the camp was moderate. A review of the activities done during the camp shows that the camp focused more on code writing and not analyzing existing code. In our future

camps, we plan to bring code analysis more into focus in the curriculum.

5. Discussion

5.1 Motivation

Based on the results of our survey, no significant changes were observed in several aspects of students' motivation, such as career interest, interest in computing, the value of learning computing, perception of their critical thinking skills, and perception of proficiency in coding. However, students' perception of proficiency in coding did increase slightly. This finding is not surprising. Most students started the camp with basic understanding of programming. However, the snake game activity required them to integrate both their prior programming knowledge as well as the new programming knowledge taught at the camp for their app. Interestingly, we did not observe significant changes in the other motivation subscales across time. It may be possible that the short duration of the camp precluded students from fully exploring the possibilities of programming and the applicability of programming in their current lives.

5.2 Programming Knowledge Analysis

Based on the overall pre-and post- total score, our project-based curriculum for the weeklong programming camp was effective in increasing programming knowledge. This aligns with the expectations set by other studies which have used project-based approaches in two-week programming camps (Franklin et al., 2013) and other longer informal learning opportunities (Wang et al., 2019). The snake game allowed students in our programming camp to incrementally learn programming concepts while making progress and seeing their game come to fruition. Since the snake game required the usage of several core programming concepts, such as variables, basic operations, loops, data structures, conditional statements, and functions, students had to learn how to implement them for their snake game to work. For example, the students needed to know what purpose variables served in the snake game, such as an integer that kept track of scores, which instructors covered during the camp.

Testing for computational thinking skills in an informal education setting, like a programming camp, is very seldomly done in research (Tang et al., 2020). Although we did not explicitly test for all individual computational thinking skills directly, we were able to group questions based on three types of computational thinking skills, such as basic abstraction concepts, analysis/debugging of existing code, and algorithmic/logical thinking by original code construction (Tang et al., 2020). The programming camp was successful in increasing overall knowledge of core programming concepts; however, the results of our programming knowledge pre-and post- scores with grouped questions showed that there are some computing skills, such as code analysis and debugging, where students had some trouble answering.

The following sub-sections will discuss the role of computational thinking skills covered using the results of different sub-group question types such as basic abstraction concepts, code analysis/debugging, and original code construction.

5.2.1 Basic Abstraction Concepts

Students were asked to answer basic variable and variable manipulation questions using math operations. In terms of teaching the basic programming concepts, such as variables, the increase of understanding in these concepts covered in these questions could be attributed to the fact that students had a good starting point on how variables and operations could work within the context of the snake game. Since variables and performing mathematical and logical operations on variables is a level of abstraction K-12 students may not be familiar with, research shows that program execution or deep familiarity of the context in which these concepts will be used, such as a game, can help students visualize how these programming concepts work(Mladenović et al., 2021). Since the instructors had continuously demonstrated the snake game throughout the programming camp, students were able to make connections on how abstraction was used in creating game components, such as displaying scores, updating snake tail length, and changing values for their game customizations on the fly. For teaching students about basic abstraction concepts, like variables, helping students visualize their final project outcome by demonstrating the game can support their learning of programming concepts. Although visualization in the form of Powerpoint animations, whiteboard examples, or sketching can provide support for students to learn basic abstraction concepts (Mladenović et al., 2021), our results show that demonstrating the project they will work on, like the snake game, and explicitly connecting it to programming concepts can be used as a visual aide to support their learning as well. This is aligned with several empirical studies which emphasized rich, visual coding experiences for students to learn basic abstraction concepts (Tang et al., 2020).

5.2.2 Code Analysis and Debugging

The second category of questions required students to analyze and debug code. Between pre-and post- scores for this group, developing this computational thinking skill did not change significantly. The lack of change between pre-and post- scores could be the fact that the instructors did not explicitly ask students to analyze pre-existing code or teach the specific code scenarios, such as the undefined variables and starting a while loop with a met condition. Although there have been many calls-to-actions to teach K-12 students a more systematic approach to debugging and analyzing code, it is common to not extensively cover debugging strategies when the goal is to generate interest in code (Michaeli & Romeike, 2019). However, since students do face issues while coding, Debugging and analysis of code on an on-demand basis to help students fix their code is the common approach, since students may sometimes attempt debugging techniques unsuccessfully, leaving them to feel helpless when they are unable to make progress (Michaeli & Romeike, 2019). In our case, we did not have enough time to strategically teach the students systematic ways to debug their code on their own, so it is reflected in our results for this question group.

Since creating code and debugging code are different skills (Michaeli & Romeike, 2019), it probably is the case that our students did not have ample time to develop their debugging skills, especially when looking at code that was not written by them. Analyzing and debugging others' code also requires more practice and development of their debugging skills both systematically and unsystematically (Bryant et al., 2019; Wilson, 2020). Reading and understanding code that was not written by them requires training students to decipher syntax and semantic meaning of the code (Lynch et al., 2019). It requires outlining and coming up with the conceptual picture of the code's intention, which requires practice (Busjahn & Schulte, 2013). It is to no surprise that students who attend programming camps with a short duration like ours likely did not develop these advanced computing skills due to lack of time to practice in class.

5.2.3 Algorithmic and Logical Thinking Skills

Questions in this third category required students to construct original code based on a particular prompt. For example, writing a loop that prints a string five times or constructs a function that adds two integers. According to the pre-and post-scores within this question group, there was a significant increase in students' algorithmic and logical thinking skills through constructing original code. Although instructors provided pieces of code to students, students were guided through the process of constructing original code for the snake game through daily incremental progress. Creating original code based on the prompt involves the development of algorithmic thinking skills, such as defining the problem, gathering relevant and applicable concepts, thinking of the logical steps, and writing the code (Braswell, 2020; Young et al., 2017). Each day, students were tasked to complete progress on another snake game milestone, such as the snake game interface on Day 1 or game piece placements on Day 2. To complete this functionality, instructors introduced the relevant programming concepts needed to complete those game milestones for that day, such as loops and conditional statements on Day 2. Using those concepts, students constructed the next snake game milestone with instructor guidance in algorithmically thinking through the problem. Although there are not many studies in informal learning context on project-based curriculums for developing algorithmic thinking skills, there are several studies in K-12 education that show that project-based curriculum can help teach students algorithmic thinking skills (Chiazzese et al., 2018; Garneli et al., 2015; Karaman et al., 2017).

6. Limitations

Although our one-week programming camp provided opportunities to learn how to code and explore computing to middle school students, we recognize that our single study, sample size, and location may not be generalizable to other groups. This may limit potential replications of our project-based short programming camp experience. Currently, we are in the process of collecting more programming camp data to strengthen our developing findings on programming knowledge and motivation to learn coding.

Secondly, our programming knowledge assessment reflected our curriculums' content coverage, but we realize that it may not have been appropriate to test students to analyze and debug code during the assessment since we did not intentionally cover it during the camp. In the next programming camp, we plan to either simplify our programming camp assessment questions to include and cover simpler forms of code analysis and debugging and/or remove these questions.

Thirdly, we recognize that the participants self-selected to participate in the programming camp and students already came in with some basic understanding. Maintaining interest after sparking initial interest increases the

likelihood of future pursuit of a related career (Christensen et al., 2014; Hidi & Renninger, 2006; Hirsch et al., 2017; Taub et al., 2012); however, we must be careful to not generalize by saying that the programming camp was successful in promoting interest where no interest may have existed in middle school students who attend the camp. Since they were self-selected, it is not surprising that their interest and motivation was high.

7. Conclusion

Informal learning environments, such as programming camps, can provide the opportunity to empower students to create a project from the ground up while learning basic programming concepts. However, instructors need to balance content coverage in terms of introducing other fundamental computational thinking skills, such as debugging and analyzing code. To keep students interested in the programming camp, we may need to temporarily forgo teaching them (and testing them) on more complex computational thinking skills such as reading code that was not created by them and debugging skills.

Regardless of making cuts to content coverage, a week-long project-based programming camp can inspire and teach students to code in a short amount of time. Although our programming camp did not significantly change students' attitudes towards pursuing computing due to students coming in with high interest in programming, we did significantly increase their programming knowledge and their perceptions of their ability to code, which could support their self-efficacy to jumpstart and continue exploring the tech field in high school, and, hopefully, into college.

Acknowledgments

This work was supported by the Boeing Distinguished Professorship funds granted to Dr. Olusola Adesope by Washington State University.

References

2018 NSSME+. (n.d.). *NSSME*. Retrieved April 5, 2021, from http://horizon-research.com/NSSME/2018-nssme

Adams, J. C. (2010). Scratching middle schoolers' creative itch. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 356–360. https://doi.org/10.1145/1734263.1734385

Akcaoglu, M. (2014). Learning problem-solving through making games at the game design and learning summer program. *Educational Technology Research and Development*, *62*(5), 583–600. https://doi.org/10.1007/s11423-014-9347-4

Al-Bow, M., Austin, D., Edgington, J., Fajardo, R., Fishburn, J., Lara, C., Leutenegger, S., & Meyer, S. (2009). Using game creation for teaching computer programming to high school students and teachers. *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, 104–108. https://doi.org/10.1145/1562877.1562913

Austin, K., & Pinkard, N. (2008). The organization and management of informal and formal learning. *Proceedings of the 8th International Conference on International Conference for the Learning Sciences - Volume 3*, 5–7.

Ayar, M. C., & Yalvac, B. (2016). Lessons Learned: Authenticity, Interdisciplinarity, and Mentoring for STEM Learning Environments. *International Journal of Education in Mathematics, Science and Technology*, *4*(1), 30–43.

Bagiati, A., Yoon, S. Y., Evangelou, D., & Ngambeki, I. (2010). Engineering Curricula in Early Education: Describing the Landscape of Open Resources. *Early Childhood Research and Practice ·*, *12*(2). https://files.eric.ed.gov/fulltext/EJ910909.pdf

Bell, P., Lewenstein, B., Shouse, A. W., & Feder, M. A. (2009). *Learning science in informal environments: People, places, and pursuits* (p. 12190). National Academies Press. https://doi.org/10.17226/12190

Braswell, K. M. (2020). From Camp to Conferences: Experiences in Leveraging Tech Conferences to Inspire Black and Latinx Girls to Pursue Coding and Tech Careers. *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, *1*, 1–4. https://doi.org/10.1109/RESPECT49803.2020.9272429

Bryant, C., Chen, Y., Chen, Z., Gilmour, J., Gumidyala, S., Herce-Hagiwara, B., Koures, A., Lee, S., Msekela, J., Pham, A. T., Remash, H., Remash, M., Schoenle, N., Zimmerman, J., Dahlby Albright, S., & Rebelsky,

S. A. (2019). A Middle-School Camp Emphasizing Data Science and Computing for Social Good. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 358–364. https://doi.org/10.1145/3287324.3287510

Burack, C., Melchior, A., & Hoover, M. (2018). *Do After-school Robotics Programs Expand the Pipeline into STEM Majors in College? (RTP)*. 2018 ASEE Annual Conference & Exposition, Salt Lake City, Utah. https://doi.org/10.18260/1-2--30341

Bureau of Labor Statistics. (2019). *Computer and Information Technology* (Occupational Outlook Handbook). U.S. Department of Labor. https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Busjahn, T., & Schulte, C. (2013). The use of code reading in teaching programming. *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, 3–11. https://doi.org/10.1145/2526968.2526969

Cabrera, R., de los Ángeles Carrión, M., & Carrión, A. (2021). Camps IEEE Ecuador: A proposal to increase children's interest in STEM areas. *2021 IEEE XXVIII International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, 1–4. https://doi.org/10.1109/INTERCON52678.2021.9532982

Chaudhary, V., Agrawal, V., Sureka, P., & Sureka, A. (2016). An Experience Report on Teaching Programming and Computational Thinking to Elementary Level Children Using Lego Robotics Education Kit. *2016 IEEE Eighth International Conference on Technology for Education (T4E)*, 38–41. https://doi.org/10.1109/T4E.2016.016

Chiazzese, G., Arrigo, M., Chifari, A., Lonati, V., & Tosto, C. (2018). Exploring the Effect of a Robotics Laboratory on Computational Thinking Skills in Primary School Children Using the Bebras Tasks. *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, 25–30. https://doi.org/10.1145/3284179.3284186

Christensen, R., Knezek, G., Tyler-Wood, T., & Gibson, D. (2014). Longitudinal analysis of cognitive constructs fostered by STEM activities for middle school students. *Knowledge Management and E-Learning*, *6*, 103–122.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

DeWitt, A., Fay, J., Goldman, M., Nicolson, E., Oyolu, L., Resch, L., Saldaña, J. M., Sounalath, S., Williams, T., Yetter, K., Zak, E., Brown, N., & Rebelsky, S. A. (2017a). Arts Coding for Social Good: A Pilot Project for Middle-School Outreach. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 159–164. https://doi.org/10.1145/3017680.3017795

DeWitt, A., Fay, J., Goldman, M., Nicolson, E., Oyolu, L., Resch, L., Saldaña, J. M., Sounalath, S., Williams, T., Yetter, K., Zak, E., Brown, N., & Rebelsky, S. A. (2017b). What We Say vs. What They Do: A Comparison of Middle-School Coding Camps in the CS Education Literature and Mainstream Coding Camps (Abstract Only). *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 707. https://doi.org/10.1145/3017680.3022434

Ericson, B., & McKlin, T. (2012). Effective and sustainable computing summer camps. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 289–294. https://doi.org/10.1145/2157136.2157223

Fields, D. A., Quirke, L. C., & Amely, J. (2015). Measuring learning in an open-ended, constructionist-based progamming camp: Developing a set of quantitative measures from qualitative analysis. *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 15–17. https://doi.org/10.1109/BLOCKS.2015.7368993

Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., Dreschler, G., Aldana, G., Almeida-Tanaka, P., Kiefer, B., Laird, C., Lopez, F., Pham, C., Suarez, J., & Waite, R. (2013). Assessment of computer science learning in a scratch-based outreach program. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 371–376. https://doi.org/10.1145/2445196.2445304

Frye, M. T., Nair, S. C., & Meyer, A. (2016). Evaluation of miniGEMS 2015 – Engineering Summer Camp for Middle School Girls. *2016 ASEE Annual Conference & Exposition Proceedings*, *7*.

Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015). Computing education in K-12 schools: A review of the literature. *2015 IEEE Global Engineering Education Conference (EDUCON)*, 543–551. https://doi.org/10.1109/EDUCON.2015.7096023

Hidi, S., & Renninger, K. A. (2006). The Four-Phase Model of Interest Development. *Educational Psychologist*, *41*(2), 111–127. https://doi.org/10.1207/s15326985ep4102_4

Hirsch, L. S., Berliner-Heyman, S., & Cusack, J. L. (2017). Introducing Middle School Students to Engineering Principles and the Engineering Design Process Through an Academic Summer Program. *INTERNATIONAL JOURNAL OF ENGINEERING EDUCATION*, *33*(1, B), 398–407.

Hofstein, A., & Rosenfeld, S. (1996). Bridging the Gap Between Formal and Informal Science Learning. *Studies in Science Education*, *28*(1), 87–112. https://doi.org/10.1080/03057269608560085

Hugerat, M. (2016). How teaching science using project-based learning strategies affects the classroom learning environment. *Learning Environments Research*, *19*(3), 383–395. https://doi.org/10.1007/s10984-016-9212-y

Jones, S. (2019, January 8). STEM Instruction: How Much There Is and Who Gets It. *Education Week*. https://www.edweek.org/teaching-learning/stem-instruction-how-much-there-is-and-who-gets-it/2019/01

Karaman, S., Anders, A., Boulet, M., Connor, J., Gregson, K., Guerra, W., Guldner, O., Mohamoud, M., Plancher, B., Shin, R., & Vivilecchia, J. (2017). Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT. *2017 IEEE Integrated STEM Education Conference (ISEC)*, 195–203. https://doi.org/10.1109/ISECon.2017.7910242

Lakanen, A.-J., & Kärkkäinen, T. (2019). Identifying Pathways to Computer Science: The Long-Term Impact of Short-Term Game Programming Outreach Interventions. *ACM Transactions on Computing Education*, *19*(3), 20:1-20:30. https://doi.org/10.1145/3283070

LePendu, P., Cheung, C., Salloum, M., Sheffler, P., & Downey, K. (2020). Summer Coding Camp as a Gateway to STEM. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 1351. https://doi.org/10.1145/3328778.3372637

Lusa Krug, D., Bowman, E., Barnett, T., Pollock, L., & Shepherd, D. (2021). Code Beats: A Virtual Camp for Middle Schoolers Coding Hip Hop. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 397–403. https://doi.org/10.1145/3408877.3432424

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

Lynch, K., Hill, H. C., Gonzalez, K. E., & Pollard, C. (2019). Strengthening the Research Base That Informs STEM Instructional Improvement Efforts: A Meta-Analysis. *Educational Evaluation and Policy Analysis*, *41*(3), 260–293. https://doi.org/10.3102/0162373719849044

Maiorca, C., Roberts, T., Jackson, C., Bush, S., Delaney, A., Mohr-Schroeder, M. J., & Soledad, S. Y. (2021). Informal Learning Environments and Impact on Interest in STEM Careers. *International Journal of Science and Mathematics Education*, *19*(1), 45–64. https://doi.org/10.1007/s10763-019-10038-9

Master, A., Cheryan, S., Moscatelli, A., & Meltzoff, A. N. (2017). Programming experience promotes higher STEM motivation among first-grade girls. *Journal of Experimental Child Psychology*, *160*, 92–106. https://doi.org/10.1016/j.jecp.2017.03.013

McCombs, J. S., Rand Education (Institute), & Wallace Foundation (Eds.). (2011). *Making summer count: How summer programs can boost children's learning*. RAND.

Michaeli, T., & Romeike, R. (2019). Current Status and Perspectives of Debugging in the K12 Classroom: A Qualitative Study. *2019 IEEE Global Engineering Education Conference (EDUCON)*, 1030–1038. https://doi.org/10.1109/EDUCON.2019.8725282

Miller, K., Sonnert, G., & Sadler, P. (2018). The Influence of Students' Participation in STEM Competitions on Their Interest in STEM Careers. *International Journal of Science Education, Part B: Communication and Public Engagement*, *8*(2), 95–114. https://doi.org/10.1080/21548455.2017.1397298

Mladenović, M., Žanko, Ž., & Aglić, M. (2021). The impact of using program visualization techniques on learning basic programming concepts at the K–12 level. *Computer Applications in Engineering Education*, *29*, 145–159. https://doi.org/10.1002/cae.22315

Mohr-Schroeder, M., Jackson, C., Miller, M., Walcott, B., Little, D., Speler, L., Schooler, W., & Schroeder, D. (2014). Developing Middle School Students' Interests in STEM via Summer Learning Experiences: See Blue STEM Camp. *School Science and Mathematics*, *114*. https://doi.org/10.1111/ssm.12079

National Science Board. (2016). *Science and Engineering Indicators 2016.* (NSB-2016-1). National Science Foundation.

Nite, S. B., Bicer, A., Currens, K. C., & Tejani, R. (2020). Increasing STEM Interest through Coding with Microcontrollers. *2020 IEEE Frontiers in Education Conference (FIE)*, 1–7. https://doi.org/10.1109/FIE44824.2020.9274273

Outlay, C. N., Platt, A. J., & Conroy, K. (2017). Getting IT Together: A Longitudinal Look at Linking Girls' Interest in IT Careers to Lessons Taught in Middle School Camps. *ACM Transactions on Computing Education*, *17*(4), 20:1-20:17. https://doi.org/10.1145/3068838

Roberts, T., Jackson, C., Mohr-Schroeder, M. J., Bush, S. B., Maiorca, C., Cavalcanti, M., Craig Schroeder, D., Delaney, A., Putnam, L., & Cremeans, C. (2018). Students' perceptions of STEM learning after participating in a summer informal learning experience. *International Journal of STEM Education*, *5*(1), 35. https://doi.org/10.1186/s40594-018-0133-4

Sadler, K., Eilam, E., Bigger, S. W., & Barry, F. (2018). University-led STEM outreach programs: Purposes, impacts, stakeholder needs and institutional support at nine Australian universities. *Studies in Higher Education*, *43*(3), 586–599. https://doi.org/10.1080/03075079.2016.1185775

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Stewart, O. G., & Jordan, M. E. (2017). "Some explanation here": A case study of learning opportunities and tensions in an informal science learning environment. *Instructional Science*, *45*(2), 137–156. https://doi.org/10.1007/s11251-016-9396-7

Tai, R. H., Liu, C. Q., Maltese, A. V., & Fan, X. (n.d.). *Planning Early for Careers in Science*. 2.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, *148*, 103798. https://doi.org/10.1016/j.compedu.2019.103798

Taub, R., Armoni, M., & Ben-Ari, M. (2012). CS Unplugged and Middle-School Students’ Views, Attitudes, and Intentions Regarding CS. *ACM Transactions on Computing Education*, *12*(2), 8:1-8:29. https://doi.org/10.1145/2160547.2160551

Wang, C., & Frye, M. (2019). miniGEMS 2018 Summer Camp Evaluation: Empowering Middle School Girls in STEAM. *2019 IEEE Integrated STEM Education Conference (ISEC)*, 149–155. https://doi.org/10.1109/ISECon.2019.8881981

Wang, C., Frye, M., & Nair, S. (2019, April 5). *The Practices of Play and Informal Learning in the miniGEMS STEAM Camp*. 2018 Gulf Southwest Section Conference. https://peer.asee.org/the-practices-of-play-and-informal-learning-in-the-minigems-steam-camp

Warner, J. R., Fletcher, C. L., Torbey, R., & Garbrecht, L. S. (2019). Increasing Capacity for Computer Science Education in Rural Areas through a Large-Scale Collective Impact Model. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1157–1163. https://doi.org/10.1145/3287324.3287418

Webb, H. C., & Rosson, M. B. (2011). Exploring careers while learning Alice 3D: A summer camp for middle school girls. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 377–

382. https://doi.org/10.1145/1953163.1953275

Weinberg, A. E., Basile, C. G., & Albright, L. (2011). The Effect of an Experiential Learning Program on Middle School Students' Motivation toward Mathematics and Science. *RMLE Online: Research in Middle Level Education*, *35*(3), 1–12.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. https://doi.org/10.1145/2157136.2157200

Wilson, K. (2020). Exploring the Challenges and Enablers of Implementing a STEM Project-Based Learning Programme in a Diverse Junior Secondary Context. *International Journal of Science and Mathematics Education*. https://doi.org/10.1007/s10763-020-10103-8

Xianglei, C., & Weko, T. (2009). *Students who study science, technology, engineering, and mathematics (STEM) in postsecondary education*. National Center for Education Studies. http://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=2009161

Young, J. R., Ortiz, N., & Young, J. L. (2017). STEMulating Interest: A Meta-Analysis of the Effects of Out-of-School Time on Student STEM Interest. *International Journal of Education in Mathematics, Science and Technology*, *5*(1), 62–74.

Zamin, N., Ab Rahim, H., Savita, K. S., Bhattacharyya, E., Zaffar, M., & Katijah Mohd Jamil, S. N. (2018). Learning Block Programming using Scratch among School Children in Malaysia and Australia: An Exploratory Study. *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, 1–6. https://doi.org/10.1109/ICCOINS.2018.8510586

Zweben, S., & Bizot, B. (2020). *Total Undergrad CS Enrollment Rises Again, but with Fewer New Majors; Doctoral Degree Production Recovers From Last Year's Dip*. 61.

Appendix A

Pre- and Post-Knowledge Assessment Administered

Boeing Programming Boot Camp for Middle Scholar 2019

1- What are these variables (integer, float, or string)?

n = 10

x = 0.98

s = 'dog'

ANSWER: n is an Integer; x is a float, s is a string

2- What is the result of code below?

20-2(3+5)*

ANSWER = 4

3- What is the result of code below?

x = 10

y = 30

z = 400

*z - y * x*

ANSWER = 100

4- what would this code print?

```
while n > 10:
    print(n)
    n = n+1
```

ANSWER = it prints nothing

5- what dose the code below prints?

```
jar = ['candy', 'gums', 'm&m']
hungry = True
for x in jar:
    if x == 'gums':
        print('jane is happy')
    elif x == 'candy':
        print('alex is happy')
    elif x == 'm&m':
        if hungry:
            print('I need food')
        else:
```

```
        print('party time')
    else:
        print('marry is happy')
```

ANSWER:

alex is happy

jane is happy

I need real food

6- what would this code print?

```
n = 10
while n > 10:
    print(n)
    n = n+1
```

ANSWER: it prints nothing

7- Write a loop that prints 'GO COUGES!' five times:

ANSWER1:

```
n = 0
while n < 5:
    print('GO COUGS!')
    n = n+1
```

ANSWER2:

```
for i in range(5):
    print('GO COUGS!')
```

BOTH ANSWERS ARE CORRECT

8- Answer the following questions based on the below list

names = ['kris', 'aj', 'jake', 'robert', 'liz']

a- write a loop to print all the elemnts of the list x:

ANSWER1:

```
for name in names:
    print(name)
```

ANSWER2:

```
n_names = len(names)
for idx in range(n_names):
    print(names[idx])
```

ANSWER3:

```
idx = 0
n_names = len(names)
while idx < n_names:
    print(names[idx])
    idx = idx+1
```

ANY OF THESE ANSWERS ARE CORRECT

b- what is the result of the code below?

```
print(len(x))
```

ANSWER = 5

9- write a function that takes two variables in its argument and returns the addition.

ANSWER:

```
def add(n1, n2):
    return n1+n2
```

10- Write a function get the first name and last name as input and print the 'first_name last_name is awesome!'?

ANSWER:

```
def awesome(first_name, last_name):

    print(first_name + ' ' + last_name + 'is awesome!')
```

Appendix B

Samples of Low and High Rated Responses to Programming Questions

Low Rated Responses	High Rated Responses