Journal of Information Technology Education: Research

An Official Publication of the Informing Science Institute
InformingScience.org

JITEResearch.org

Volume 19, 2020

IF YOU CAN PROGRAM YOU CAN WRITE: LEARNING INTRODUCTORY PROGRAMMING ACROSS LITERACY LEVELS

Ziva R. Hassenfeld*	Tufts University, Medford, MA, USA	Ziva.Hassenfeld@tufts.edu
Madhu Govind	Tufts University, Medford, MA, USA	Madhu.Govind@tufts.edu
Laura E. de Ruiter	Tufts University, Medford, MA, USA	Laura.DeRuiter@tufts.edu
Marina Umaschi Bers	Tufts University, Medford, MA, USA	Marina.Bers@tufts.edu

* Corresponding author

ABSTRACT

Aim/Purpose	This paper presents findings on a curricular intervention aimed at integrating computer programming with reading and writing in early elementary school. The purpose of this research was to explore the relation between students' varying literacy levels and their level of success in mastering an introductory programming language.
Methodology	This curricular intervention study was implemented in a single school district in southeastern Virginia. Of the district's 33 elementary schools, eight schools received an external grant from the U.S. Department of Defense to introduce computer science in early elementary education. Standardized literacy test scores were correlated with internally developed, and age appropriate programming assessment scores from $N = 132$ second grade students.
Contribution	This study is the first of its kind to look at how students at varying literacy levels succeed in mastering an introductory programming language when introduced through a literacy lens.
Findings	The findings indicated that there was strong evidence for a weak, positive correlation between students' literacy levels, as determined by the PALS assessment, and their programming mastery, as determined by the curricular programming assessments. The positive correlation suggests that there may indeed be underlying constructs that overlap between literacy and programming.

Accepting Editor France Cheong | Received: November 1, 2019 | Revised: January 6, February 3, February 13, 2020 | Accepted: February 18, 2020.

Cite as: Hassenfeld, Z. R., Govind, M., de Ruiter, L., E., & Bers, M. U. (2020). If you can program you can write: Learning introductory programming across literacy levels. *Journal of Information Technology Education: Research, 19,* 65-85. https://doi.org/10.28945/4509

Recommendations for Practitioners	Consider integrating computer programming as a foundational component of the literacy curriculum, especially in the early grades, where the two skill sets can mutually support one another.
Recommendations for Researchers	Additional research is necessary, using a variety of literacy and programming measures, to continue to understand the relationship between emerging literacy and emerging computer programming skills.
Impact on Society	Reimagining computer programming as a language has significant implications for how we teach programming in schools and how students then use programming out in the workforce.
Future Research	Future work will repeat this curricular intervention with younger students: the district's first grade and kindergarten classrooms. Introducing programming through the Coding as Literacy (CAL) approach even earlier in students' literacy trajectories, we believe, will allow the positive impact of programming knowledge to influence students' literacy development. In this next phase of our research agenda, we will collect pre and post literacy scores, both standardized and internally developed, to see the myriad ways that programming knowledge impacts literacy.
Keywords	emerging programming, pedagogy, curricula, early childhood, computer science education, literacy

INTRODUCTION

Simon, a second grade student in southeastern Virginia, participated in a 12-hour robotics-based programming curriculum. The experience was different than his other classes. He had never seen a tangible programming language. The KIBO robotics kit and its accompanying wooden blocks were nothing like the screen-based programming puzzles he had done on Code.org. The curriculum combined learning of the robotics kit and the programming language blocks with a children's book. Simon was used to clear distinctions among his various subjects, so having "KIBO Time" in the middle of his typical reading block was unusual but refreshing. All of the questions his teachers asked in the lessons were open-ended, and so too were the prompts he was asked to work on in his design journal. He was continually asked to think, create, imagine and revise his work in a way that he isn't often asked to do in school. Simon is a "PALS-identified" literacy student, which means he performed below average on the Phonological Awareness Literacy Screening (PALS) assessment and thus qualifies for remedial literacy instruction. However, his writing in his design journal to document his process was clear, engaging, and sophisticated—and so were the programs he wrote for the KIBO robot. Simon, in this new learning context, asserted a very different student identity. He was far from remedial. As a composer of program and prose, he was exceptional.

Simon was part of a district-wide initiative to pilot the integration of computer science into the core curriculum at the early elementary level. Thirteen second grade classrooms (N = 132 students) participated in a 12-lesson robotics curriculum that incorporates the tangible KIBO robotics platform (see e.g., Bers, 2018b) and explicitly connects programming concepts to parallel concepts in natural language and literacy. The goal of the curriculum, called CAL, is to understand "coding as literacy" (CAL), that is, programming as a creative, meaning-making, and communicative act, as is the case with reading and writing. This pilot study, which took place in a complex district in Virginia, the first U.S. state to mandate the teaching of computer science in K-12 education, was the first to explore how students engage with programming through the CAL approach.

For decades, researchers have been interested in the correlation between programming knowledge and other academic knowledge, or, more precisely, the possibility of transfer from programming skills to other disciplines and domains (see e.g., Denning, 2017; Scherer, 2016). Recently, Scherer,

Siddiq and Sánchez Viveros (2018) published a meta-analysis of 105 studies that looked at the transfer of programming skills to a variety of domains. They found a positive transfer effect to situations that require creative thinking, mathematical skills and metacognition, but they found very little transfer effect to students' literacy skills. Scherer et al. conclude, "Reading comprehension and writing skills [are] skills that overlap only marginally with programming," (p. 783). However, the authors note that there were relatively few studies that explored the transfer effect on literacy, and those studies did not necessarily seek to foster literacy through programming.

This paper starts from a different theoretical perspective that there is significant overlap when using natural and artificial languages (Fedorenko, Ivanova, Dhamala, & Bers, 2019) and specifically, that there is overlap between writing skills and programming skills (Bers, 2019; Vee, 2013, 2017). However, these connections have largely been overlooked by the field of computer science education when developing pedagogical approaches. This paper presents findings on one such effort to integrate programming and reading and writing in early elementary school. Specifically, we look at the relation between students' varying literacy levels and their level of success in mastering an introductory programming language when taught through an intentionally integrative curriculum.

Conceptual Framework

Part of the reticence to viewing programming as connected to reading and writing is the history of separation between subject matter. The idea that subject matter is discipline-specific and is the most significant fault line upon which to organize classroom teaching and learning became popular in the 1980s (Reisman, 2012; Shulman & Quinlan, 1996). The underlying premise was that disciplines contain distinct forms of knowledge with their own unique modes of inquiry (Bruner, 1960; Hirst, 1965; Schwab, 1978), and as Reisman (2012) explained, "Experts in each field had normative definitions of domain-specific knowledge and understanding," (e.g., Schoenfeld, 1985; Shulman, 1987; Wineburg, 1991). Out of this body of research emerged the still popular idea of Pedagogical Content Knowledge (PCK), the idea that an English teacher needs familiarity with the canon of English literature, literary theory, and, equally significant, knowledge of how to scaffold this body of knowledge and set of skills for young students (Shulman, 1987). In other words, learning to read and think like a historian (Wineburg, 2001) is different than learning to read and think like a scientist.

A decade later, the field of education experienced an uptick of interest in integrated curriculum (Czerniak, Weber, Sandmann, & Ahern, 1999). Multiple national reform efforts popped up, stressing the importance of making connections across the curriculum (see e.g., National Council of Teachers of English [NCTE], 1996; National Council of Teachers of Mathematics [NCTM], 1989; National Research Council [NRC], 1996). Drake and Burns (2004) write about three forms of integrated curriculum, of which the curricula themselves often don't distinguish: inter-disciplinary, multi-disciplinary, and trans-disciplinary. Inter-disciplinary refers to a curriculum focused on a particular skill and applied to various disciplines (e.g., working on sourcing in science, English and biology). Multi-disciplinary refers to a curriculum that chooses one topic to explore in a variety of disciplines (e.g., studying apples in literature, history, and chemistry). Trans-disciplinary refers to a curriculum focused on an issue and not particular subjects (e.g., studying food scarcity in America and allowing that to touch economics, psychology, child development, etc.).

Programming, an emerging discipline in elementary education, became associated with science and technology and has been integrated with STEM (Science, Technology, Engineering and Mathematics) curricula (Bers, 2019; Clements, 1999; Guzdial & Morrison, 2016). The assumed wisdom is that the auxiliary skills for programming are math and science, not literacy. This has led to the creation of robotics and programming applications that are based on solving challenges with increased complexity, which leave out the creative and self-expressive aspects of programming that align more closely with literacy. More recently, programming has been integrated with the arts and social sciences to fill these gaps (Aguirre-Muñoz & Pantoya, 2016; Maguth, 2012; Sullivan & Bers, 2017; Sullivan, Strawhacker,

& Bers, 2017). These types of integration, however, still do not address how literacy can be fostered through programming, hence the creation of the Coding as Literacy (CAL) approach.

The CAL Approach: Beyond Integration

The CAL approach, developed by Bers (2019), says programming is much more aligned with literacy than ever before. CAL's premise is that programming is a language, that is, a "system of communication, natural or artificial, composed of a formal system of signs, governed by syntactic and grammatical combinatory rules, that serves to communicate meaning by encoding and decoding information" (Bers, 2019, p. 64) and, as such, it should be taught in tandem with literacy. CAL is premised on a re-imagination of what programming in early childhood is, itself, as a discipline.

To understand our re-imagination, it is important to return to Seymour Papert and the Constructionist movement. Seymour Papert proposed that the true power of computer science education was providing students with a new medium for expression and communication through the production of technological artifacts (Bers, 2018b; Kafai & Resnick, 1996; Papert, 1980). In other words, programming as a language, like natural language—is a tool for expression and communication. This, they argued, needed to be emphasized.

The CAL approach returns to Papert's conception of programming as a tool for thinking in new ways and expressing these ideas through the use of a computer language. In a recent article, Hassenfeld and Bers (2020) articulated how writing and programming mirror one another as compositional activities. We argued that writing and coding are both compositional processes that share a sub-set of activities: planning and prewriting, creating and drafting, testing and evaluating, debugging and editing/revising.

Planning and Prewriting. In coding, *planning* refers to the process the programmer goes through before she begins programming the actual composition or project. It can involve creating a flowchart or completing a design journal (Strawhacker & Bers, 2015). In writing, *prewriting* refers to everything that happens before the writer begins the actual composition. It can involve doing research on a topic, outlining one's composition, and planning the prose one wishes to write.

Creating and Drafting. In coding and writing, both creating and drafting refer to the actual creation of the composition or project. The writer writes the words on the page and the programmer uses the icons or text of the programming language.

Testing and Evaluating. In both coding and writing, testing and evaluating are the review process. The programmer watches her program run and the writer reads her composition. This is a chance to see whether the composition accomplishes what the writer/programmer had planned. To do so the writer must become the reader and the programmer must become the viewer and observe the results of the computer's compiler.

Debugging and Editing and Revising. Debugging in coding, like editing and revising in writing, is the activity of fixing one's existing composition. This can involve mechanical errors (editing), for example, the programmer forgot a start block in her programming composition and so the character did not move, or the writer started a sentence without a capitalized letter. It can also involve stylistic changes (revising). For example, the programmer changes her program to have a character move fewer steps or the writer adds particular details to her written composition to help clarify. In both cases, the composer reviews her composition, identifies the gaps, and then engages the debugging/editing and revising process to address those gaps.

The fact that both compositional activities, writing and coding, involve these four sub-activities suggests that the integration of coding into literacy, especially in the early grades where both skillsets are still being developed, is essential.

For the most part, while the field of literacy has embraced technological tools produced *by* coding, the field has yet to consider the relationship between coding and literacy (Vee, 2013; 2017). In other

words, while the world of literacy education understands that coding tools can be used to reinforce key existing strategies from reading and writing (e.g., summarizing, re-telling, visualizing, composing/story-telling, and sequencing), the connection is understood to be one of support, not mutual reinforcement (see e.g., https://www.literacyworldwide.org/blog/digital-literacies/teaching-with-tech).

The CAL approach comes at a time where there is a rising call for thoughtful curriculum in early childhood programming. In a recent review of the research, Buitrago Flórez et al. (2017) wrote:

> It is important to mention that there is a critical gap in terms of educational research focused on teaching and learning CS and programming. Despite the significant number of articles written by computer scientists that investigate issues related to teaching and learning CS and programming, research conducted by experts in the field of education are almost nonexistent in most countries. Because social and cultural factors have a direct impact in learning, they must be taken into account in curricula design. **Thus, constructivist, sociocultural, and pedagogical approaches are needed to create curricula that are geared toward the development of CT skills in primary, middle, and high school. These skills will, without a doubt, be fundamental for the vast majority of jobs in the 21st century (p. 852).**

The CAL-KIBO curriculum begins to fill this gap. The curriculum takes programming outside of STEM and reframes it as a language. The curriculum is centered around children's books and has students create interpretations and text-to-world connections through open-ended, tangible programming projects and written compositions.

The CAL-KIBO Curriculum

In order to introduce programming as a language, to be used for communication and expression, the CAL-KIBO curriculum explicitly connects each programming concept to parallel concepts in natural language and literacy.

The CAL-KIBO curriculum (CAL-K) consists of 12 one-hour lessons oriented around the popular children's book *Where the Wild Things Are* by Maurice Sendak. Topically, the lessons are designed to introduce students to the KIBO programming language, beginning with the easier programming blocks and then moving into the more difficult programming blocks. Before introducing any programming blocks, there are a number of foundational activities in the first two lessons that establish programming as a language. One activity (Lesson 1: Programmers and Writers) visually presents the Design Process next to the Writing Process and has the students discuss the similarities and differences between programmers and writers. Another activity (Lesson 2: Tools of Communication) has the students play a game of telephone first with whispering, then with a handwritten message and finally with a typed message. The students explore how the canvas/medium of communication impacts the clarity of the message. For example, typed messages are easier to decipher than handwritten messages for some readers but maybe not for others. Computers are also readers of code but very different readers than humans.

After these introductory lessons, the curriculum begins introducing the KIBO robot and its programming blocks. As the programming blocks are introduced, the curriculum weaves between what the new blocks can do and the connections back to natural language. This connection is mediated in the curriculum through the focus book, *Where the Wild Things Are*. For example, one activity (Lesson 6: What Did Max Sense) reviews the way the protagonist of the book, a little boy named Max, uses his five senses in the story. This activity leads into the introduction of the various sensors and sensor blocks in the KIBO programming language and robotics kit. The teachers are guided to lead a discussion comparing the "poetic" language used in the story to describe Max's senses and the contrasting command language needed for sensor blocks in the KIBO programming language. Another activity (Lesson 8: Repeated Loops) looks at how repetition is used as a literary device in *Where the Wild Things Are* and compares and contrasts that to the efficiency of using repeat blocks in the KIBO pro-

gramming language. These sorts of activities are augmented with other activities that support students in understanding programming as a language. For example, debugging is introduced as the programming corollary to revising. Just as writing doesn't always perfectly convey our message to the reader, our programs don't always perfectly convey our intended message to the robot. We revise and debug to perfect our communication.

The final project in the curriculum has students create a "Wild Rumpus" in both writing and programming. In the book *Where the Wild Things Are*, the Wild Rumpus takes up six wordless pages. The students are invited to imagine what was happening in the Wild Rumpus and add their own ideas as to what might be fun to do during a Wild Rumpus. The students describe the Wild Rumpus in writing and then try to program their KIBO to do it. Between compositional activities, after writing and before programming, the students engage in a meta-cognitive reflection on the constraints and affordances of expression in natural language and programming. To accomplish this, the students participate in a technology circle where the teacher poses the following questions: Are there certain activities that you wrote about that you can code with KIBO? Are there certain activities you wrote about that might not work with KIBO? How will you change your idea so that it makes sense for KIBO?

Over the course of the curriculum, students gain a foundational understanding of programming concepts such as algorithmic thinking, debugging and control structures and discuss how they are analogous to literacy concepts such as story structure, awareness of audience and tools of communication, among others. By the end of this introductory programming curriculum, algorithmic thinking is connected to sequencing in stories, debugging is connected to revision, and control structures are linked to cause and effect in storytelling. Both writing and programming are framed for the children as tools for self-expression and creative communication.

KIBO Programming Language

The curriculum described here utilizes KIBO robotics and its tangible programming language. The KIBO robotics kit is a block programming language created by the DevTech Research Group through funding from the National Science Foundation. KIBO involves hardware (the robot itself) and software (tangible programming blocks) used to make the robot move (Sullivan & Bers, 2015). KIBO, as a beginner programming language, is explicitly designed to meet the developmental needs of young children. The kit contains easy-to-connect construction materials including: wheels, motors, light output, and a variety of sensors. KIBO's language is displayed on interlocking wooden programming blocks (see Figure 1). These wooden blocks contain no embedded electronics or digital components. Instead, KIBO has an embedded scanner in the robot. This scanner allows users to scan the KIBO robot with sensors and light output attached. No computer, tablet, or other form of "screen-time" is required to learn programming with KIBO.

KIBO's programming language contains a total of 21 different individual programming blocks, with many increasingly complex programming concepts that can be introduced such as repeat loops, conditional statements and nesting statements. As a block-based programming language, KIBO supports the act of programming by "encoding the grammar of the language" into the individual blocks using specific color and shape attributes (Weintrop & Wilensky, 2015, p. 200). For instance, every program begins with a green "begin" block, and a blue block with a left arrow icon corresponds to a "turn left" action. Assembling and scanning a sequence of KIBO blocks introduces children to the central ideas of programming in a way that is playful, tangible, and easy to use.

Figure 1. KIBO Robotics Kit

Tangible programming interfaces like KIBO are important tools for learning for young children. Research on educational robotics indicate that children show cognitive gains by being able to physically manipulate blocks and visualize their thought processes; behavioral gains by showing greater motivation for learning; and socioemotional gains by working collaboratively with peers to assemble their robotic creations (Horn & Bers, 2018). Recent research on KIBO has shown that beginning in pre-kindergarten, children are able to master foundational sequencing concepts using KIBO's programming language (Sullivan & Bers, 2015, 2017; Sullivan, Strawhacker, & Bers, 2017). This study also found that as children got older, they were able to master more complex concepts such as repeat loops and conditional statements (Sullivan & Bers, 2015). In addition to these robotic and programming components, the KIBO kit also contains art platforms that can be used for children to personalize their projects with crafts materials.

KIBO was designed to be developmentally appropriate and to be used alongside curriculum. In the aforementioned section, we described the scope and sequence of the CAL-KIBO curriculum and how students engage with the KIBO robotics kit through the CAL pedagogical approach. In these subsequent sections, we describe the pilot study in which the CAL-KIBO curriculum was implemented.

Research Question

This pilot study is part of a larger research agenda into CAL that involves several dimensions: 1) the creation of programming environments, such as KIBO robotics and ScratchJr explicitly designed with a literacy approach (Bers, 2018a), 2) curricular materials for emergent and early readers; 3) a pedagogical approach with professional development strategies that explicitly highlight the connection between the activity of programming and the mastering of a natural language and its uses to convey meaning, 4) classroom studies to understand the affordances of this approach compared to other approaches, and 5) experimental studies in lab settings to characterize cognitive mechanisms using fMRI techniques to explore if the language networks in the brain activate when programming.

In this paper, we will explore one such classroom study in a complex district in Virginia, specifically focusing on how students of different literacy levels succeeded with the CAL curriculum. We asked the following research question: Among the second grade students who participated in the CAL-KIBO curriculum, did the students at higher literacy levels, as measured by a standardized literacy assessment, gain greater mastery of the programming curriculum than the students at lower literacy levels?

METHOD

PARTICIPANTS/CLASSROOM CONTEXT

This study was implemented in a single school district in southeastern Virginia with a high percentage of military-connected students. Of the district's 33 elementary schools, eight schools received an external grant from the U.S. Department of Defense to introduce computer science in early elementary education. We focused this research on four of these eight schools because they followed similar training and implementation schedules. Teachers from these four schools attended the same one-day training in the fall and subsequently implemented the CAL-KIBO curriculum in the spring. The analytic sample for this pilot study was N = 132 students (M_{age} = 7.61 years, SD = 0.38) from 13 classrooms (and thus, 13 second grade teachers). Only students whose teachers implemented the curriculum with adequate fidelity and who participated in both literacy and programming assessments were included in the analytic sample. For further details on our standards for inclusion, see Appendix A.

TEACHER TRAINING AND SUPPORT

Participating teachers agreed to implement the curriculum twice a week (two lessons, approximately one hour per lesson) for six weeks between February and March 2019. None of the teachers had prior programming experience or were familiar with the CAL approach before this study. All participating teachers attended a full-day training in the fall semester, which introduced teachers to the CAL approach, the KIBO robotics kit, and the scope and sequence of the CAL-KIBO curriculum. Participating teachers also engaged in ongoing professional learning throughout implementation, including video tutorials for every lesson, coaching calls with our team, and additional in-person support from the schools' instructional technology resource technicians (ITRTs). ITRTs participated in the same training as teachers and attended two additional days of professional learning, conducted by the research team, on how to observe classrooms and provide support for teachers.

DATA SOURCES

PHONOLOGICAL AWARENESS AND LITERACY SCREENING (PALS)

Developed by the University of Virginia Curry School of Education and supported by the Virginia Department of Education, PALS (https://pals.virginia.edu/) is the state-provided screening tool for the Virginia Early Intervention Reading Initiative (EIRI). The purpose of EIRI is "to reduce the number of children with reading problems by detecting those problems through early diagnosis and immediate intervention," (Introduction and Overview, p. 3). This assessment looks at phonological awareness, alphabet knowledge, letter-sound knowledge, phonetic spelling, concept of word, and word recognition. Tasks and items were selected "because of their standing in literacy research, their technical properties, and their correlation to the Commonwealth of Virginia's Standards of Learning," (Introduction and Overview, p. 3). Through PALS, students who need additional literacy instruction/intervention services are identified. The PALS assessment score used in this research was administered in late November 2018.

Reliability of the PALS assessment is reported in the instrument's technical manual (Invernizzi, Sullivan, & Meier, 2004). Average internal consistency estimates from the total pilot sample are acceptable (Cronbach's α = .83). Inter-rater reliability was reported to be stable for all tasks (r = .99).

KIBO MASTERY CHALLENGES (KMC)

The KIBO Mastery Challenges (KMCs) were used to gain insight into students' understanding of the KIBO programming language. The KMCs, a second iteration of the Solve-Its, utilized widely in previous studies (e.g., Sullivan & Bers, 2018; Sullivan, Bers, & Mihm, 2017; Sullivan, Strawhacker, &

Bers, 2017), were embedded in the CAL-KIBO curriculum at four different points (at the end of lessons 4, 6, 8 and 12). They assessed students' mastery of the KIBO programming language and programming concepts taught up to that point in the curriculum (See Appendix B for more details about how the KMCs aligned with the curriculum). Each of the four KMCs was comprised of six multiple choice items with a single correct answer choice and several incorrect distractor options. The questions were phrased in an age-appropriate language. Classrooms that implemented all four KMCs were included in the analytic sample (See Appendix A for more details on inclusion criteria).

During the administering of the KMCs, the questions were presented in the same order, and no feedback was provided for any question. Aligning with the increasing difficulty of subsequent lessons, the later assessments were designed to be harder than the earlier ones due to the content of the curriculum. In order to ascertain precise question difficulty, we used item response theory to calculate each question's difficulty index. The difficulty index was used to calculate a weighted score for each question reflective of this population, with more weight being given to more difficult questions. Once weighted, each student's individual KMC scores were summed to obtain a single composite score.

DATA ANALYSIS

LITERACY AND PROGRAMMING

To investigate the relationship between students' KMC composite scores, i.e., their mastery of the introductory programming language curriculum and their literacy ability (as measured by the PALS score), we used Bayesian linear regression using the BayesFactor package version 0.9.12-2 (Morey & Rouder, 2015) in R (R Core Team, 2016). Details of the implementation are provided below. With Bayes factors, several statistical models can be directly compared with each other, and the strength of the evidence for each model can be determined. The analysis tells us under which statistical model our data are most likely. Researchers sometimes use verbal labels to describe the strength of the evidence that the Bayes factors provide. Table 1 gives an overview of a common textual interpretation of Bayes factor values.

Table 1. Common Textual Interpretation of Bayes Factor Values, adapted from Jeffreys (1961), cited in Wetzels et al. (2011). H_A = alternative hypothesis, H_0 = null hypothesis

BAYES FACTOR	INTERPRETATION
> 100	Decisive evidence for H_A
30 – 100	Very strong evidence for H_A
10 – 30	Strong evidence for H_A
3 – 10	Substantial evidence for H_A
1 – 3	Anecdotal evidence for H_A
1	No evidence
1/3 – 1	Anecdotal evidence for H_0
1/10 – 1/3	Substantial evidence for H_0
1/30 – 1/10	Strong evidence for H_0
1/100 – 1/30	Very strong evidence for H_0
< 1/100	Decisive evidence for H_0

Linear regression from the BayesFactor package allows for the incorporation of random factors such as students. In our analysis, we are interested in the potential effect of literacy scores on students' KMC scores, but not in the individual variance associated with each student. Including student as a random factor allows making inferences about the role of the main effect that we are interested in (literacy scores) that do not depend on a particular student. Similarly, we also included teacher as a random factor, as potential teacher effects are not the focus of this pilot study.

Using the generalTestBF function in the BayesFactor package, we specified a full model with KMC composite scores as the dependent variable, and PALS scores, gender, TeacherID, and StudentID as independent variables (with TeacherID and StudentID specified as random factors). The function successively removes terms from the full model and tests the resulting submodels, which can then be compared with each other. In line with recommendations by Morey and Rouder (2011), we used a Cauchy prior with scale parameter $1/\sqrt{2}$ for the standardized effect size.

RESULTS

DESCRIPTIVES

KMC composite scores for the $N = 132$ students ranged from 1.57 to 6.24 ($M = 3.36$, $SD = 1.08$). The distribution of scores is shown in Figure 2. PALS scores ranged from 7-68 ($M = 46.58$, $SD = 14.33$). The distribution of the scores is shown in Figure 3.

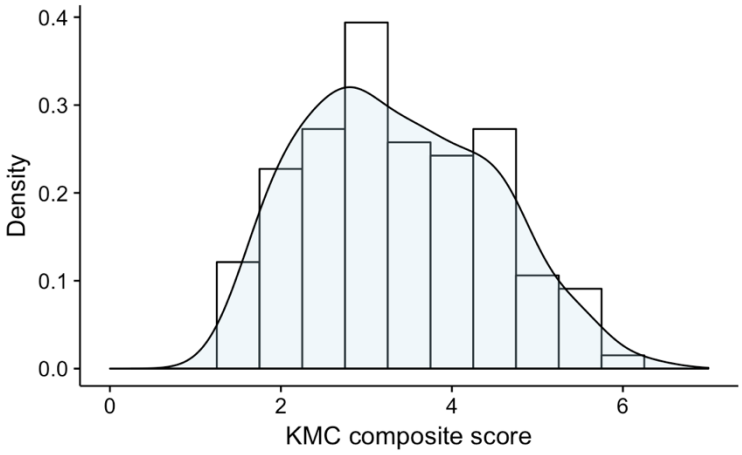

Figure 2. Histogram of KMC composite scores with overlaid density plot ($N = 132$).

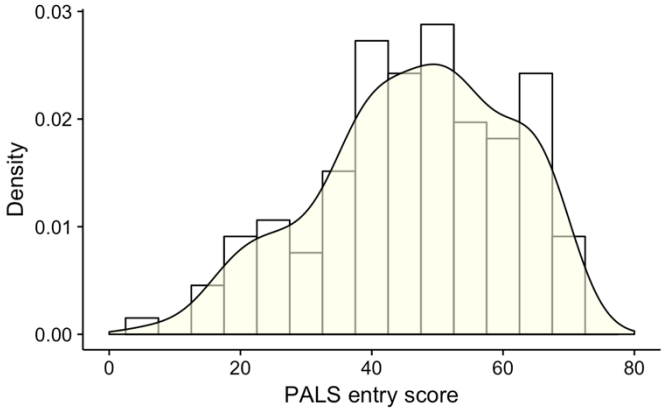

Figure 3. Histogram of PALS entry scores with overlaid density plot ($N = 132$).

Inferential Statistics

The Bayesian regression determined that the preferred model contained the PALS entry score and the random factors StudentID and TeacherID (but not gender). Comparing this model directly with the model that does not include the PALS scores (i.e., a model with only the random effects StudentID and TeacherID), we obtain a Bayes factor of 4.11. This means that the data are more than four times more likely under the model with PALS scores than the one without, which is considered substantial evidence.

We followed up on the relationship between KMC composite scores and PALS scores with Bayesian correlations, using the jzs_cor function in the BayesMed package (Nuijten, Wetzels, Matzke, Dolan, & Wagenmakers, 2015). The correlation between the two variables was 0.3, with a Bayes factor of 43.85 (very strong evidence). Figure 4 shows the relationship between students' KMC composite scores and their PALS score.

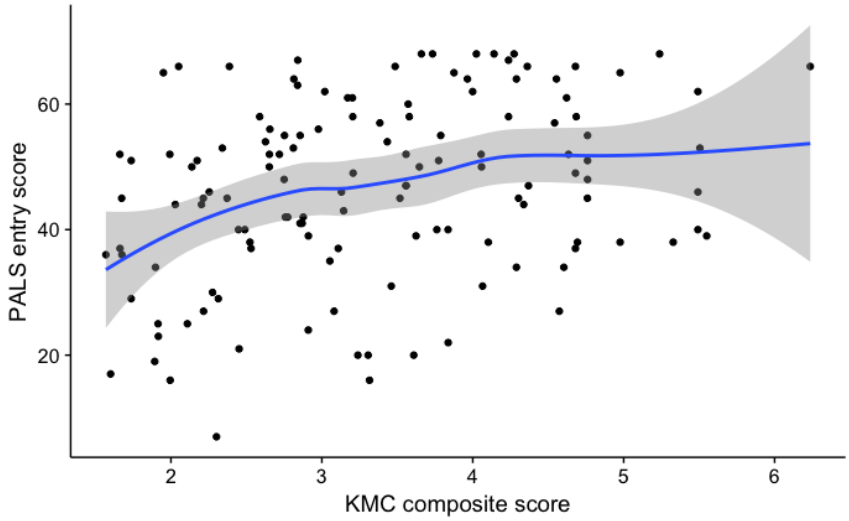

Figure 4. Scatterplot showing the relationship between KMC composite scores and PALS entry scores. The blue line indicates the smoothed conditional mean, the grey area the associated confidence bands.

Discussion

The CAL-KIBO curriculum was designed to introduce young students to programming as a language, analogous to a natural language in its expressiveness and open-endedness within a pre-set system of rules and vocabulary. In this pilot study we sought to investigate whether students at higher literacy levels would also perform higher in programming, compared to students at lower literacy levels. We suspected a correlation based on our conceptual framework of programming as distinct but akin to natural language, but we did not know how strong the correlation would be.

The findings indicated that there was strong evidence for a weak, positive correlation between students' literacy levels, as determined by the PALS assessment, and their programming mastery, as determined by the KMCs. The size of the correlation is in line with previous related research on the relationship between computational thinking and verbal ability in older children and adolescents. Román-González, Peréz-González, and Jiménez-Fernández (2017) found a weak positive correlation (.273) between a computational thinking test and the verbal factor of the Primary Mental Abilities Battery (Thurstone, 1938) in 10- to 16-year-olds. While our study is not fully comparable to this study, both indicate that language/literacy ability has some role to play when it comes to mastering age-appropriate computational concepts.

The positive correlation suggests that there may indeed be underlying constructs that overlap between literacy and programming, which future experimental studies will explore more closely. However, the correlation was not so strong that lower literacy students were not able to perform well on the KMCs. In other words, while the findings suggest a shared underlying construct between literacy and programming that promotes students with strong literacy skills to more readily master programming skills, the relationship is not deterministic. The pedagogical implications of this are significant.

The fact that students from all literacy levels were able to master an introductory programming language, albeit, with varying levels of ease, can be leveraged for the development of literacy. Recent research (Hassenfeld & Bers, 2020; Thompson, Tanimoto, Berninger, & Nagy, 2016) has shown how programming composition can support young students in writing composition. Specifically, facility and comfort with debugging can help students with the cognitively and conceptually difficult work of revision in writing and enhance their motivation to do so. Furthermore, prior research has shown that immersion in a computer programming language can lead to increases in language mechanics, reading comprehension, and verbal creativity (Clements, 1999; Studyvin & Moninger, 1986). If students at all literacy levels are able to master age-appropriate programming, then young children who are learning foundational reading and writing skills can use their understanding of programming as a resource for their writing development and other literacy skills.

There are a number of limitations inherent in the design of this study. Most significantly, we don't know the causality of the correlation. We don't know whether the weak, positive correlation between literacy levels and mastery of the KIBO programming language and programming concepts, as taught by the curriculum, was because of the language-based nature of the curriculum, i.e., introducing programming as a language akin to natural language and related to literacy, or if the correlation is simply an expression of general mental ability (e.g., Spinath, Spinath, Harlaar, & Plomin, 2006), that is, that strong students often do well in school in all curricular areas. We are also missing students' experience of the curriculum in their own voice. How did the stronger and weaker literacy students experience the curriculum? Is there a difference in how they might self-report on the difficulty of the lessons? We plan to address these limitations in the next phase of this research.

Simon, introduced in the opening vignette, is a "PALS-identified" literacy student. He is below grade level in all of the literacy markers. And yet, in the CAL-KIBO curriculum, he not only scored highly in the KMCs, evidencing mastery of the curriculum and introductory programming language, but his writing embedded in the curriculum was also strong. We have two hypotheses for why this might be the case. First, taking literacy outside of the formal literacy block allowed Simon to revisit reading and writing anew, free of his remedial literacy identity (McDermott, 2010; Steele, Spencer, & Aronson, 2002). If so, developmentally appropriate programming has the potential to serve as a motivator for students struggling with reading and writing. Second, and more significantly, learning an introductory programming language through the CAL approach reinforced essential concepts for literacy, i.e., juxtaposing algorithms in programming with sequencing and story structure in literacy. Perhaps the practice of scanning syntactically correct and incorrect sequences of KIBO blocks allowed Simon to transfer his understanding of algorithms into literacy as he composed writing reflections in his design journal with clear and sophisticated story structure. In our future work we hope to identify the underlying skills of programming that permeate into literacy and the extent to which this transfer may occur.

Future work will repeat this curricular intervention with younger students: the district's first grade and kindergarten classrooms. Introducing programming through the CAL approach even earlier in students' literacy trajectories, we believe, will allow the positive impact of programming knowledge to influence students' literacy development. In this next phase of our research agenda, we will collect pre and post literacy scores, both standardized and internally developed, to see the myriad ways that programming knowledge impacts literacy. For example, mastering a block based introductory programming language may not impact students' spelling inventory, but learning the powerful idea of algorithms in computer science, i.e., that programs are constructed using step-by-step instructions,

may impact students' sequencing abilities in reading, thereby improving their capacity to summarize, predict and ask questions, all key reading strategies. The CAL pedagogical approach, as evident from these findings, has the potential of fostering literacy through programming.

CONCLUSION

This study is the first of its kind to look at how students at varying literacy levels succeed in mastering an introductory programming language when introduced through a literacy lens. The findings indicated that there was strong evidence for a weak, positive correlation between students' literacy levels, as determined by the PALS assessment, and their programming mastery, as determined by the curricular programming assessments. The positive correlation suggests that there may indeed be underlying constructs that overlap between literacy and programming. Educators should consider integrating computer programming as a foundational component of the literacy curriculum, especially in the early grades, where the two skillsets can mutually support one another. Additional research is necessary, using a variety of literacy and programming measures, to continue to understand the relationship between emerging literacy and emerging computer programming skills.

REFERENCES

Aguirre-Muñoz, Z., & Pantoya, M. L. (2016). Engineering literacy and engagement in kindergarten classrooms. *Journal of Engineering Education*, *105*(4), 630-654. https://doi.org/10.1002/jee.20151

Bers, M. U. (2018a). Coding as a literacy for the 21st century. *Education Week*. Retrieved from http://blogs.edweek.org/edweek/education_futures/2018/01/coding_as_a_literacy_for_the_21st_century.html

Bers, M. U. (2018b). *Programming as a playground: Programming and computational thinking in the early childhood classroom*. London and New York: Routledge Press. https://doi.org/10.4324/9781315398945

Bers, M. U. (2019). Programming as another language: Why computer science in early childhood should not be STEM. In C. Donohue (Ed.), *Exploring key issues in early childhood and technology: Evolving perspectives and innovative approaches* (pp. 63-70). https://doi.org/10.4324/9780429457425-11

Bruner, J. S. (1960). *The process of education*. Cambridge, MA: Harvard University Press.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834-860. https://doi.org/10.3102/0034654317710096

Clements, D. H. (1999). The future of educational computing research: The case of computer programming. *Information Technology in Childhood Education*, *1999*(1), 147–179. Association for the Advancement of Computing in Education (AACE). Retrieved from https://www.learntechlib.org/p/10815/

Czerniak, C. M., Weber, W. B., Sandmann, A., & Ahern, J. (1999). A literature review of science mathematics integration. *School Science and Mathematics*, *99*(8), 421-430. https://doi.org/10.1111/j.1949-8594.1999.tb17504.x

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33-39. https://doi.org/10.1145/2998438

Drake, S. M., & Burns, R. C. (2004). *Meeting standards through integrated curriculum*. Alexandria, VA: Association for Supervision and Curriculum Development.

Fedorenko, E., Ivanova, A., Dhamala, R., & Bers, M. U. (2019). The language of programming: A cognitive perspective. *Trends in Cognitive Sciences*, *23*(7), 525–528. https://doi.org/10.1016/j.tics.2019.04.010

Guzdial, M., & Morrison, B. (2016). Growing computer science education into a STEM education discipline. *Communications of the ACM*, *59*(11), 31–33. https://doi.org/10.1145/3000612

Hassenfeld, Z. R., & Bers, M. U. (2020). Debugging the writing process: Lessons from a comparison of students' coding and writing practices. *The Reading Teacher*. https://doi.org/10.1002/trtr.1885

Horn, M., & Bers, M. U. (2018). Tangible computing. In S. A. Fincher, & A. V. Robins (Eds.), *The Cambridge handbook of computing education research* (pp. 663-678). Cambridge, England: Cambridge University Press. https://doi.org/10.1017/9781108654555.023

Hirst, P. (1965). Liberal education and the nature of knowledge. In R. D. Archambault (Ed.), *Philosophical analysis and education* (pp. 113 - 138). London: Routledge.

Invernizzi, M., Sullivan, A., & Meier, J. (2004). *Phonological Awareness Literacy Screening for Preschoolers (PALS-PreK).* Charlottesville, VA: University of Virginia. https://doi.org/10.1037/t27727-000

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford: Clarendon Press.

Kafai, Y., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world.* Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Maguth, B. M. (2012). In defense of the social studies: Social studies programs in STEM education. *Social Studies Research and Practice*, *7*(2), 65-90. Retrieved from http://www.socstrpr.org/wp-content/uploads/2012/08/MS06393-5.pdf

McDermott, R. P. (2010). The acquisition of a child by a learning disability. In S. Chaiklin & J. Lave (Eds.), *Understanding learning: Influences and outcomes* (pp. 60-70). Cambridge, England: Cambridge University Press. https://doi.org/10.1017/CBO9780511625510.011

Morey, R. D., & Rouder, J. N. (2011). Bayes factor approaches for testing interval null hypotheses. *Psychological Methods*, *16*(4), 406–419. https://doi.org/10.1037/a0024377

Morey, R. D., & Rouder, J. N. (2015). *BayesFactor: Computation of Bayes factors for common designs*. R Packages on CRAN. Retrieved from https://rdrr.io/cran/BayesFactor/

National Council of Teachers of English. (1996). *NCTE / IRA standards for the English Language arts*. Newark, Delaware & Urbana, Illinois: International Reading Association, & National Council of Teachers of English. Retrieved from https://ncte.org/resources/standards/ncte-ira-standards-for-the-english-language-arts/

National Council of Teachers of Mathematics. (1989). *Curriculum and evaluation standards for school mathematics.* Reston, VA: The Council. Retrieved from http://csmc.missouri.edu/PDFS/CCM/summaries/standards_summary.pdf

National Research Council. (1996). *National science education standards*. Washington, DC: National Academies Press. https://doi.org/10.17226/4962

Nuijten, M. B., Wetzels, R., Matzke, D., Dolan, C. V., & Wagenmakers, E. J. (2015). *BayesMed: Default Bayesian hypothesis tests for correlation, partial correlation, and mediation*. R Packages on CRAN. Retrieved from https://rdrr.io/cran/BayesMed/

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

R Core Team (2016). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Reisman, A. (2012). Reading like a historian: A document-based history curriculum intervention in urban high schools. *Cognition and Instruction*, *30*(1), 86-112. https://doi.org/10.1080/07370008.2011.634081

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Scherer, R. (2016). Learning from the past – The need for empirical evidence on the transfer effects of programming skills. *Frontiers in Psychology*, *7,* 1390. https://doi.org/10.3389/fpsyg.2016.01390

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2018). The cognitive benefits of learning programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, *111*(5), 764-792. https://doi.org/10.1037/edu0000314

Schoenfeld, A. H. (1985). *Mathematical problem solving*. Cambridge, MA: Academic Press.

Schwab, J. J. (1978). The practical: A language for curriculum. In I. Westbury, & N. J. Wilkof (Eds.), *Science, curriculum, and liberal education: Selected essays* (pp. 287–321). Chicago, IL: University of Chicago Press.

Shulman, L. (1987). Knowledge and teaching: Foundations of the new reform. *Harvard Educational Review*, *57*(1), 1-23. https://doi.org/10.17763/haer.57.1.j463w79r56455411

Shulman, L., & Quinlan, K. (1996). The comparative psychology of school subjects. In D. Berliner & R. Calfee (Eds.), *Handbook of educational psychology* (pp. 399-437). New York, NY: Simon & Schuster, Inc.

Spinath, B., Spinath, F. M., Harlaar, N., & Plomin, R. (2006). Predicting school achievement from general cognitive ability, self-perceived ability, and intrinsic value. *Intelligence*, *34*(4), 363-374. https://doi.org/10.1016/j.intell.2005.11.004

Steele, C. M., Spencer, S. J., & Aronson, J. (2002). Contending with group image: The psychology of stereotype and social identity threat. *Advances in Experimental Social Psychology*, *34*, 379-440. https://doi.org/10.1016/S0065-2601(02)80009-0

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education, 25*(3), 293-319. https://doi.org/10.1007/s10798-014-9287-7

Studyvin, D., & Moninger, M. (1986). Logo as an enhancement to critical thinking. Paper presented at *Meeting of the Logo 86 Conference*, Cambridge, MA.

Sullivan, A., & Bers, M. U. (2015). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, *26*(1), 3-20. https://doi.org/10.1007/s10798-015-9304-5

Sullivan, A., & Bers, M. U. (2017). Dancing robots: Integrating art, music, and robotics in Singapore's early childhood centers. *International Journal of Technology and Design Education*, *28*(2), 325-346. https://doi.org/10.1007/s10798-017-9397-0

Sullivan, A., & Bers, M. U. (2018). The impact of teacher gender on girls' performance on programming tasks in early elementary school. *Journal of Information Technology Education: Innovations in Practice*, *17,* 153-162. https://doi.org/10.28945/4082

Sullivan, A., Bers, M. U., & Mihm, C. (2017). Imagining, playing, & programming with KIBO: Using KIBO robotics to foster computational thinking in young children. In S. C. Kong, J. Sheldon, & K. Y. Li (Eds.). *Proceedings of the International Conference on Computational Thinking Education* (pp. 110-115). Wanchai, Hong Kong: The Education University of Hong Kong. Retrieved from https://www.eduhk.hk/cte2017/doc/CTE2017%20Proceedings.pdf#page=121

Sullivan, A., Strawhacker, A., & Bers, M. U. (2017). Dancing, drawing, and dramatic robots: Integrating robotics and the arts to teach foundational STEAM concepts to young children. In M. S. Khine (Ed.), *Robotics in STEM education: Redesigning the learning experience* (pp. 231-260). Springer, Cham. https://doi.org/10.1007/978-3-319-57786-9_10

Thompson, R. H., Tanimoto, S. L., Berninger, V. W., & Nagy, W. (2016). Programming, reading, and writing: Integrated instruction in written language. In *Proceedings of 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 73-77). Cambridge, UK: IEEE. https://doi.org/10.1109/VLHCC.2016.7739667

Thurstone, L. L. (1938). *Primary mental abilities*. Chicago, IL: University of Chicago Press.

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, *1*(2), 42-64. https://doi.org/10.21623/1.1.2.4

Vee, A. (2017). *Programming literacy: How computer programming is changing writing*. Cambridge, MA: The MIT Press. https://doi.org/10.7551/mitpress/10655.001.0001

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)* (pp. 199–208). https://doi.org/10.1145/2771839.2771860

Wetzels, R., Matzke, D., Lee, M. D., Rouder, J. N., Iverson, G. J., & Wagenmakers, E. J. (2011). Statistical evidence in experimental psychology: An empirical comparison using 855 *t* tests. *Perspectives on Psychological Science*, *6*(3), 291–298. https://doi.org/10.1177/1745691611406923

Wineburg, S. (1991). Historical problem solving: A study of the cognitive processes used in the evaluation of documentary and pictorial evidence. *Journal of Educational Psychology*, *83*(1), 73-77. https://doi.org/10.1037/0022-0663.83.1.73

Wineburg, S. (2001). *Historical thinking and other unnatural acts: Charting the future of teaching the past*. Philadelphia, PA: Temple University Press.

APPENDIX A

Inclusion Criteria

From the four elementary schools, 333 total students from 15 classrooms participated in the CAL-KIBO curriculum. Included in the final dataset were only those classrooms that met the project's standards for inclusion, detailed in the chart below. The analytic sample for this study was $N = 132$ students ($M_{age} = 7.61$ years, $SD = 0.38$) from 13 classrooms. Sex was split almost evenly: 49.2% female and 50.8% male. The majority of students (87.9%) were identified as Native English speakers and 3.0% as Dual Language Learners. Chi-square tests revealed no significant demographic differences in sex, number of Native English speakers, and number of Dual Language Learners between the analytic and full samples.

COMPONENT	DESCRIPTION	CRITERION
Classroom observations	Instructional technology specialists (ITRTs) observed classrooms at least three times over the course of implementation using the Positive Technological Development (PTD) Checklist for Environment and Facilitators (see e.g., Bers, 2018b).	At least three observations were recorded. Teachers who met inclusion standards received moderate-to-high observation scores.
Teacher self-report	Teachers participated in surveys, interviews and lesson logs in order to capture their perceptions and experiences with the curriculum. • Surveys were administered online at four time points (pre/post training and pre/post curriculum). • Interviews were conducted either in-person or via conference call at five time points (pre/during training and pre/mid/post curriculum). • Lesson logs, completed by teachers at the end of every lesson, captured teachers' highlights and challenges during the lesson and how they may have modified the lesson to meet their classroom's needs.	Teachers participated in multiple surveys, interviews, and lesson logs and displayed adequate adherence to the CAL-KIBO curriculum.
Student data	Students actively participated in the curriculum, evidenced by their design journals and KIBO Mastery Challenges (KMCs). • Student design journals consisted of writing prompts and programming reflections at the end of every lessons. • KMCs were administered four times at the end of every few lessons (see Appendix A).	Classroom was included if the majority of students had completed prompts in design journals and participated in all four KMCs.

APPENDIX B

KIBO Mastery Challenges - Match to Curriculum

The **CAL-KIBO curriculum** is accessible using this link: https://sites.tufts.edu/programmingaslit-eracy/kibo-readers/. Embedded within the curriculum are four six-item **KIBO Mastery Challenges (KMCs)**, which assess students' mastery of the KIBO programming language and the programming concepts taught up to that point in the curriculum.

ITEM NUMBER	TYPE OF QUESTION	CURRICULAR CONCEPT & REFERENCE
1A	Programming Concept	Engineers follow a series of steps called the Design Process to turn their ideas into projects that are shared with others. *(Lesson 1, Activity: What is an Engineer, p. 18-19)*
2A	Programming Concept	Both the Design and Writing Processes are cycles - there's no official starting or ending point. You can begin at any step, move back and forth between steps, or repeat the cycle over and over. *(Lesson 1, Activity: Engineers and Writers, p. 18-19)*
3A	Programming Concept	A KIBO robot and computer are most alike because they are both machines. A pig and a house are not machines. A bike may look like a KIBO (both have wheels), but a bike is not a programmable machine. *(Lesson 2, Activity: Characteristics of Robots, p. 21)*
4A	KIBO Hardware/Software	The KIBO body is a part of the KIBO robot. *(Lesson 3, Meet the KIBO Robot, p. 26)*
5A	Programming Concept	All robots use computers to perform a function (e.g., movement) and have computers in them. Computers are not programmed to perform physical tasks and therefore are not robots. *(Lesson 2, Characteristics of Robots, p. 21)*
6A	KIBO Syntax	Each KIBO program begins with the green "Begin" block and ends with the red "End" block. *(Lesson 4, Program the Hokey-Pokey, p. 28)*
1B	Evaluate the KIBO Program	A program is a sequence of instructions that the robot performs in order. To make KIBO spin twice, the correct program sequence is "Begin, Spin, Spin, End". *(Lesson 2, Activity: Human Language vs. Code Language, p. 23*)
2B	Evaluate the KIBO Program	Debugging is a method that describes how people find errors in their programs and use different strategies to solve the problem. The green "Begin" block is missing at the start of this program. *(Lesson 5, Why is KIBO Confused, p. 31)*
3B	KIBO Hardware/Software	KIBO's Sound Sensor is shaped like an ear and senses sounds from the environment using the "Wait for Clap" block. *(Lesson 6, KIBO Sound Sensor, p. 34-35)*

ITEM NUMBER	TYPE OF QUESTION	CURRICULAR CONCEPT & REFERENCE
4B	Identify the KIBO Program	In programming, an event is an action that causes something to happen. In order for KIBO to shake only after it hears a clap, the correct program is "Begin, Wait for Clap, Shake, End". *(Lesson 6, KIBO Sound Sensor, p. 34-35)*
5B	KIBO Hardware/Software	The Sound Recorder module, like any other sensor, can be inserted into any of the four ports on the KIBO body. *(Lesson 6, KIBO Sound Recorder, p. 36)*
6B	KIBO Hardware/Software	The Sound Recorder module has three different buttons—square, triangle, and circle—which correspond to three orange Sound Recorder blocks: "Play ▢", "Play △", and "Play ◯". *(Lesson 6, KIBO Sound Recorder, p. 36)*
1C	Evaluate the KIBO Program	Parameters are used to tell the robot how many times to repeat, or when to stop repeating. In this program, the red light repeats twice. *(Lesson 8, KIBO Repeat with Number, p. 42)*
2C	Identify the KIBO Program	KIBO will only repeat commands that are placed inside of the "Begin Repeat" and "End Repeat" blocks. *(Lesson 8, KIBO Repeat with Number, p. 42)*
3C	Programming Concept	Before exploring repeat loops, students examine different KIBO programs and identify repeating patterns. In this question, the pattern is that each program has one additional "Beep" block than the program before it. *(Lesson 8, KIBO Repeat with Number, p. 41)*
4C	Evaluate the KIBO Program	The "Play △" block is placed inside a loop that repeats twice. Another "Play △" block is placed after the repeat loop, so KIBO plays the sound a total of three times. This question assesses students' understanding that each block, even if repeated, corresponds to a single action. *(Lesson 8, KIBO Repeat with Number, p. 41)*
5C	Evaluate the KIBO Program	KIBO will only repeat commands that are placed inside of the "Begin Repeat" and "End Repeat" blocks. The "Play ◯" block is placed after the repeat loop, so KIBO will not play the sound forever. *(Lesson 8, KIBO Repeat with Number, p. 42)*
6C	KIBO Syntax	A repeat loop is comprised of the "Begin Repeat" and "End Repeat" blocks. *(Lesson 8, KIBO Repeat with Number, p. 42)*
1D	KIBO Hardware/Software	KIBO uses the Distance Sensor (which is shaped like a telescope) to sense how near or far KIBO is from other objects. *(Lesson 9, KIBO Repeat with Light/Distance Sensor, p. 44)*

ITEM NUMBER	TYPE OF QUES-TION	CURRICULAR CONCEPT & REFERENCE
2D	Identify the KIBO Program	This question assesses students' understanding of repeat loops, in which the repeated action is placed inside the "Begin Repeat" and "End Repeat" blocks, and their understanding of the Light Sensor parameter ("Until Dark"). *(Lesson 9, KIBO Repeat with Light/Distance Sensor, p. 44)*
3D	Identify the KIBO Program	A sensor used with a repeat loop will continuously check for a change in its environment, but a sensor used with an if statement will check only one time. This scenario requires the use of an if statement. *(Lesson 10, KIBO If Statements, p. 47)*
4D	KIBO Syntax	A nested statement is a repeat/if statement inside of another repeat/if statement. The inner loop refers to the set of instructions bound by a "Begin Repeat/If" and "End Repeat/If". *(Lesson 10, Extended Activity: Nested Statements, p. 48-49)*
5D	Identify & Evaluate KIBO Programs	This question assesses students' understanding of repeat loops with number and nested loops. *(Lesson 10, Extended Activity: Nested Statements, p. 48-49)*
6D	Identify & Evaluate KIBO Programs	This question assesses students understanding of if statements and nested if statements. *(Lesson 10, Extended Activity: Nested Statements, p. 48-49)*

BIOGRAPHIES

Dr. Ziva R. Hassenfeld earned her doctorate in curriculum and teacher education from Stanford University in 2016. She is currently a post-doctoral fellow at the DevTech Research Group of Tufts University. Her research focuses on the tools and reading strategies young children employ when reading texts, as well as the pedagogies teachers use to support student textual interpretation, fluency, and comprehension.

Madhu Govind, MA is a doctoral student at the DevTech Research Group in the Eliot-Pearson Department of Child Study & Human Development at Tufts University. Madhu received her BS in Child Studies and Neuroscience from Vanderbilt University and her MA in Child Study & Human Development at Tufts University. Madhu's research interests include collaborative family programming and teacher perceptions of robotics and coding education in early childhood.

Professor Laura de Ruiter is a Research Assistant Professor at the DevTech Research Group at Tufts University. She holds an MSc. in Developmental Linguistics from the University of Edinburgh (UK) and a PhD in Linguistics from the Max Planck Institute for Psycholinguistics in Nijmegen (The Netherlands). Laura is a quantitative researcher who studies how children learn language. Her research ranges from topics such as how children use intonation in storytelling to how they understand complex sentences in different languages. Recently, she is investigating the relationship between children's coding skills and their language abilities.

Professor Marina Umaschi Bers is Professor and Chair, Eliot-Pearson Department of Child Study and Human Development; with a secondary appointment in Computer Science Department; Director, DevTech Research Group, Tufts University; Director, Early Childhood Technology (ECT) Graduate Certificate Program, Tufts University; Co-Founder and Chief Scientist at KinderLab Robotics; and Author of *Coding as a Playground: Computational Thinking in the Early Childhood Classroom* (2018), *Designing Digital Experiences for Positive Youth Development: From Playpen to Playground* (2012), and *Blocks to Robots: Learning with Technology in the Early Childhood Classroom* (2007).