Accepted Manuscript

Assessing elementary students' computational thinking in everyday reasoning and robotics programming

Guanhua Chen, Ji Shen, Lauren Barth-Cohen, Shiyan Jiang, Xiaoting Huang, Moataz Eltoukhy

PII: S0360-1315(17)30049-0

DOI: 10.1016/j.compedu.2017.03.001

Reference: CAE 3140

To appear in: *Computers & Education*

Received Date: 20 September 2016

Revised Date: 28 February 2017

Accepted Date: 1 March 2017

Please cite this article as: Chen G., Shen J., Barth-Cohen L., Jiang S., Huang X. & Eltoukhy M., Assessing elementary students' computational thinking in everyday reasoning and robotics programming, *Computers & Education* (2017), doi: 10.1016/j.compedu.2017.03.001.

Running Head: Assess Computational Thinking

Assessing Elementary Students' Computational Thinking in Everyday Reasoning and Robotics Programming

Guanhua Chen[a], Ji Shen[a] *, Lauren Barth-Cohen[b], Shiyan Jiang[a], Xiaoting Huang[c], Moataz Eltoukhy[a],

[a] University of Miami
[b] University of Utah
[c] Peking University

* Corresponding Author: Ji Shen, Department of Teaching and Learning, University of Miami, 5202 University Drive, Coral Gables, FL 33124

Assessing Elementary Students' Computational Thinking in Everyday Reasoning and Robotics Programming

Abstract

Based on a framework of computational thinking (CT) adapted from Computer Science Teacher Association's standards, an instrument was developed to assess fifth grade students' CT. The items were contextualized in two types of CT application (coding in robotics and reasoning of everyday events). The instrument was administered as a pre and post measure in an elementary school where a new humanoid robotics curriculum was adopted by their fifth grade. Results show that the instrument has good psychometric properties and has the potential to reveal student learning challenges and growth in terms of CT.

Keywords: computational thinking; robotics education; evaluation methodologies; programming and programming languages

1. Introduction

1.1 Background

Recent years have witnessed an increasing emphasis on integrating computer science into K-12 settings. This boom is being driven by many factors including economic and technological demands for a future workforce with necessary computer skills. In the United States, there has been a governmental level push in computer science education (Smith, 2016). Starting computer science education in early grades contributes critically to this initiative since children's early experiences likely factor into their persistence in the domain and future career choices (Margolis et al., 2010; Yardi & Bruckman, 2007). Unfortunately, there is limited research targeting elementary students in this area.

The XXX project (name omitted for blind review) is one of the many initiatives in bringing robotics and computer science curriculum to elementary students. The project utilizes the humanoid robot platform NAO by Aldebaran Robotics[1], as it contains many of the sophisticated tools such as voice/speech and face recognition, which is reflective of the current status of professional robotics and aligned with our goal of exposing students to robotics platforms that are similar to what is used by professionals. More importantly, this platform has a visual programming platform (i.e., drag-drop) that is appropriate for elementary age students. Furthermore, the NAO platform includes a robotics simulator, thus allowing the students to run their code quickly on the simulator while sharing the physical robot resulting in the cost being relatively affordable.

This paper focuses on the assessment aspect of the project. As in many other similar projects (e.g. Weintrop et al., 2014), we are deeply concerned with what students learn in this curriculum. One thing is certain, that if the only thing students take away from this curriculum is a set of specific commands that ask the robot to move forward, turn left, say "hello", etc. (which is important), this project could hardly be recognized as a success. Computer science is such a fast-changing domain that those who possess a single set of skills tied to specific software or hardware may quickly lag behind. What we deem more important is to assist our students to acquire certain skills and thinking patterns that are readily transferable, and thus conducive, to their future learning and problem solving in computing related subjects or even everyday reasoning. In order to prepare themselves for future learning and transferring (Bransford & Schwartz, 1999), students need to be equipped a set of necessary skills to make sense of the learning context from multiple angles. In this sense computational thinking emerged as our central construct for our assessment effort.

1.2 Computational Thinking

Computational Thinking (CT) was proposed by Wing (2006) as a general term that "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." It entails a whole set of mental tools that enable people to reduce difficult problems into readily solvable subtasks, represent problems appropriately, interpret data, compose algorithms that are executable by a machine, and take correctness, efficiency, and even aesthetics into consideration when solving a problem. Wing (2006) deemed CT as an essential skill that is established by the widespread application of computing and computers, just as the 3 R's (reading, writing, and arithmetic) that were facilitated by the advent of printing. Since CT has gained wide currency among scholars and practitioners,

[1] For more information about this platform, please refer to https://www.aldebaran.com/en/cool-robots/nao

it has been interpreted in many different ways (Aho, 2012; Barr & Stephenson, 2011; Cuny et al, 2010; Grover & Pea, 2013; National Research Council, 2010). For instance, Cuny et al. (2010) argued, "CT is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." This definition is further simplified by Aho (2012) as formulating problems in a way that could be solved by "computational steps and algorithms." Barr and Stephenson (2011), aiming to develop a way to define CT in K-12 settings, argued that "CT is an approach to solving problems in a way that can be implemented with a computer." CSTA (2011) proposed an operational definition of CT which dissects CT into six dimensions: (1) formulating problems in a way that machines can help to solve, (2) processing data in a logical way, (3) representing data abstractly, (4) algorithmizing the automated solutions, (5) solving problems in an efficient way, and (6) transferring knowledge and skills in solving other problems. Diverse as these proposed definitions are, CT in its essence entails at least thinking in a way that can be represented and processed by machines to formulate and solve problems. Furthermore, beyond the growing interest in CT in the research community, CT is also becoming popular among K-9 educators (Mannila et al., 2014).

1.3 Assess Computational Thinking

CT is the central construct we used to conceptualize our assessment. Through the curriculum we teach students how to program a robot to complete certain tasks (e.g., problem solving in terms of moving a robot) and examine their CT abilities in this context. In our assessment we stress the transfer (Bransford & Schwartz, 1999; Bransford et al, 2000; Haskell, 2001) component of CSTA's definition: i.e., students need to demonstrate improved CT in solving problems in other contexts, such as commanding other machines to perform similar tasks, or reasoning about everyday scenarios. The focus on transfer, we believe, is more aligned with a broad interpretation of CT - a fundamental skill for everyone.

Existing work has often focused on assessing student created artifacts for CT skills in a variety of settings. For instance, Koh et al. (2010) developed a real-time CT assessment system that stresses semantic analysis of student-created games or simulations and visualizes students' learning in terms of CT patterns. With a common focus on games, Werner et al. (2012) tested students' (10-14-year-olds) CT learning by implementing three challenges in a 3D gaming environment powered by Alice (http://www.alice.org/index.php) and examined several factors (parental education, mother languages, high school grades, etc.) and their relationships to students' CT performance. In the Scratch (https://scratch.mit.edu) platform, Seiter and Foreman (2013) proposed a CT assessment framework and demonstrated its efficacy by applying it to 150 Scratch projects done by students from grade one through six. Similar work was also done by Brennan and Resnick (2012), who developed a framework to analyze young Scratchers' CT development by looking into their artifacts. Finally, closest to the current project, Bers et al. (2014) evaluated children's (4.9-6.5-year-olds) written programs after each activity of their curriculum to determine the students' CT learning patterns.

As shown in the above examples, the majority of existing CT assessments focus more on examining student products, after they have learned a particular programing platform. This limitation prevents such assessment method from being used as pre/post measure of a specific curriculum. Furthermore, given an interpretation of CT as a fundamental skill that can be transferred across platforms, we should aim for assessment tools that apply across platforms. This is especially important given the proliferation of many coding and robotics platforms for the

elementary level (e.g. Programmable Bricks, Creative Hybrid Environment for Robotic Programming, VEX Robotics Design System, and Dash) and a need for assessment tools that cut across platforms.

Given these goals, recent work by Tew and Guzdial (2011) on assessing introductory computer science concepts is readily applicable across platforms. Their work showed the possibility of adopting a pseudo-code approach in assessing programming and thus evaluating pedagogical effectiveness of different courses. Furthermore, their work introduced a viable process of developing and validating assessment items of its kind. However, their assessment was not designed for use in elementary school and did not address the broader construct of CT, which presents as a limitation when aiming to understand if students can transfer CT in different contexts – a critical component that gives CT much of its currency. Another increasingly popular tool to assess CT is Bebras, which pools multiple challenges composed of short tasks aiming to promote informatics and CT among students of various age groups (Dagienė & Stupuriene, 2016; Duncan & Bell, 2015). While Bebras shares our goal of assessing CT given students' limited prior knowledge about computation, it does not have a robotics focus. Given these limitations, we developed a CT assessment instrument that targets elementary students and minimizes the requirement of being familiar with specific computing platform.

Based on the operational definition of CT proposed by CSTA (see section 1.2), which enjoys a wide popularity from CS teachers as well as researchers, we worked out a five-component framework (Table 1) that guided our assessment items development. We did not include the component on transfer from the CSTA framework because all of our items are designed for transfer - either in the context of solving a problem in a new coding environment that is not covered in the curriculum or in the context of reasoning about everyday scenarios. Furthermore, since fifth grade students have limited programming experience, we used a pseudo-code approach in the coding items with two forms of syntax, text-based and drag-drop.

Table 1

Computational thinking components.

S	Formulating problems and solutions using machine recognizable **syntax**
D	Organizing and analyzing **data**
A	Conceptualizing and generating solutions through **algorithms** (a series of ordered steps)
R	**Representing** problems and solutions through multiple external means such as a model and a formula
E	Generating, revising, and evaluating solutions with the goal of achieving the most **efficient** and **effective** combination of steps and resources

Given our design rationale, we have the following research questions that guided this study: What are the psychometric properties of the instrument we develop assuming it measures one

construct (i.e., CT)? How do students perform on the assessment in light of the CT dimensions in our framework, in the two transfer contexts, and in the text-based and drag-drop coding environments?

2. Methods

2.1 Robotics curriculum

We developed the robotics curriculum and implemented it in a Title one public elementary school in a southeast city in the United States. We provided a three-day teacher professional development workshop in summer before implementation, which covered the basics of the robotics platform and the major topics in the curriculum. We then implemented this curriculum for six month long starting in fall, with each weekly session lasting for between 45 and 60 minutes. During the session, each student worked on his or her own laptop using the visual programming platform. And due to the cost, only one physical robot was used. Thus, students had to write the program and test it on a virtual robot first, and then take turns to run their tested programs on the physical robot. Except for the initial and final sessions, our curriculum was organized around specific topics that are common to humanoid robotics, such as basic actions (say words, waving hands, sit down, etc.), voice recognition, tactile sensors, walking motion, and animation. Furthermore, several key computer science concepts were introduced along the way, such as algorithms, variables, conditionals, loops, serial execution, and multitasking. Four mini projects as well as a final project were included in the curriculum. Designed to help students make connections between robotics and everyday settings, these projects involved programming the robot to carry out tasks such as conducting a conversation, moving toward the destination, and performing a dance.

2.2 Participants

The school enrolled in 767 students that year (Gender: 48.8% female and 51.2% male; Ethnicity: 40.8% Black, 24.8% Hispanic, 23.1% White, 6.9% Asian, 0.4% Indian, and 4.0% two or more ethnicities). Their entire fifth grade (Six classrooms; n=125) adopted the curriculum. Due to different needs of each class, the teachers had the autonomy in terms of curriculum pacing and completion.

We administered the instrument described in the next subsection in paper/pencil as a pretest in early fall to all six fifth grade classrooms. A total of 121 students took it. The test was not timed and it took students about 20-50 minutes to finish. We report the results of pretest from all the six classes. We administered the posttest in March in one class (Class I) with 22 students and the same posttest in may in another class (Class II) with 15 students. Again, the test was not timed. It took the students about 30-60 minutes to finish. We only report the pre/post comparison for those two classes in the paper (both classes were able to complete the curriculum; the other classes, due to standardized testing and other technical issues, were not able to complete the curriculum within the academic year). In both classes I and II, each student was randomly assigned to one of two forms targeting two different programming environments (text-based or visual programming; see the next subsection for details) in the pretest and the student took the same form in the posttest.

2.3 Instrument and item design

Using our CT framework, we developed an instrument that contains 23 items organized into 6 item sets. Among the items are 15 multiple-choice questions and 8 open-ended questions. Table 2 shows the specific items for each component (note that an item may target one or more CT

components).

Table 2
Items targeting each CT component.

Component	S	D	A	R	E
Explanation	Students answer the question using correct syntax	Students notice and correctly use the data given in the question	Students come up with algorithm that correctly solves the problem	Students represent solutions in multiple ways that are consistent with each other	Students solve problem in an efficient manner
Everyday scenarios		1.1-1.4 2.1-2.3 3.3 4.1-4.3	1.1-1.4 2.1-2.2 3.1-3.3 4.1-4.3	1.3,1.4 2.3 3.1-3.2	1.2 4.2-4.3
Robotics programming	5.5 6.3, 6.5	5.1-5.4 6.1-6.3, 6.5	5.3, 5.5 6.3, 6.5	5.2-5.5 6.1-6.2, 6.4	5.5

Two different CT problem contexts were incorporated in our assessment: everyday scenarios (item sets 1-4) and robotics programming (item sets 5-6). The everyday scenarios include daily activities that could both relate to CT skills and be relatable to our fifth grade students, such as washing clothes (see Fig. 1 for an example) and being driven to school.

==Insert Figure 1 about here==

The robotics-programming items ask students to demonstrate their CT abilities using predefined commands and syntax to program a robot or a robotic arm to accomplish certain tasks (see Fig. 2 for an example).

For items in the robotics-programming context, we further prepared two forms of predefined programming languages (Fig. 2) while keeping problems identical: text-based programming language (similar to most professional programming languages) and visual programming language (or drag-drop programming, similar to Scratch). We made this arrangement since the robotics course adopts a visual programming environment. Different programing environments might affect student's understanding of certain concepts related to programming (Lewis, 2010), and one of our aims was to examine whether students would show differences in responding to items that are text-based or visual programming.

==insert Fig. 2. about here==

The instrument was reviewed by the research team to ensure the specific items address relevant CT components, a computer science faculty member to validate the computer science content, and six fifth grade teachers to ensure the readability by their students. It was piloted by a voluntary fifth grade student before it was administered as pretest.

2.4 Scoring of the Assessment

The multiple-choice items were scored dichotomously: 1 for correct and 0 for incorrect responses. For the open-ended items, we developed a set of coding rubrics in the following manner. We first identified the CT components each item was assessing and described the expected performance with respect to each component. We then constructed a three-level rubric for each CT component of each item: A response that matched the expected performance would receive 2 points; a response that partially matched the expected performance would receive 1 point; and a response that did not match the expected performance would receive 0 point. We also attached sample student responses in the rubric to illustrate the different levels. Fig. 3 shows an example of the expected performance, the corresponding rubric, and student sample responses for all the involving CT components of item 5.5. This item asks students to use given commands to draw a shape that is most similar to a triangle. Similar to this one, many items corresponded to several CT components and in those cases there would be multiple three-level rubrics for that item, one rubric corresponding to each CT component.

== insert Fig. 3 about here==

== insert Fig. 4 about here==

After we developed the specific rubrics, two researchers coded all the open-ended items in the pretest. Fig. 4 shows the iterative process of ensuring interrater reliability. In brief, the two raters first sat together to rate a sample of 18 responses to the same item (the same component), trying to reach agreement in coding upon the rubric. They then rated another 18 responses separately. If the inter-rater reliability (percentage of agreement) reached a consistency level higher than 85%, then the two raters scored the remaining responses independently after reaching agreement on the different scores of the first 18 responses. If the inter-rater reliability did not reach the expected consistency level of 85%, the two raters first discussed the inconsistency and reached agreement. Then they chose another sample of 18 responses and repeated the process until the inter-rater reliability reached the satisfactory level. The same raters sat together to code all the collected posttests.

2.5 Statistical and Content Analyses

As the test contains six testlets, or item sets (each item set has multiple items which share a common stimulus), the pretest results were analyzed using a Rasch testlet model (Wang & Wilson, 2005). This model is an extension of the simple Rasch model that takes into consideration testlet effects and yields more precise estimations of test reliability as well as item and person parameters. Testlet effect refers to the situation when a few items share the same context or stimuli, also called item stem. These items become a testlet, where items are correlated to some extent, and the possibility of answering each item correctly is not independent of the others. It is not appropriate to use a simple Item Response Theory (IRT) model, as the model assumes item local independence. Wainer and Wang (2000) used traditional IRT model and the testlet model to analyze 86 TOEFL items, and they found that the traditional approach yielded biased item parameter estimates when ignoring the testlet effect. In our analysis, six item sets are grouped into six testlets correspondingly. An existing IRT program, ConQuest (Wu et al., 1998), was used for data calibration. Using students' estimated abilities derived from the model, appropriate inferential statistical tests (e.g., t-test) were applied to determine the performance between the two forms of tests (text-based and drag-drop) and among the six classes of students

in pretest. Because of the small sample collected, we simply used total scores to run inferential statistical tests for the pre/post comparison for the two focus classes and subgroups within each class (e.g., students who had high and low pretest scores). We also examined student responses in light of our CT framework to identify themes or patterns.

==insert Fig. 5 about here==

3. Results and Discussion

3.1 Pretest

In this section, we report results from the pretest. Recall that all six classes took the pretest.

3.1.1 Psychometric properties and student performance comparison

Overall, the instrument shows promising psychometric properties based on the pretest results. The item difficulty level spans a wide range from -1.911 to 1.269 in logit scale. Estimated student abilities have a spread that matches the item difficulties very well on the scale (see Wright Map in Fig. 5). The whole instrument has an EAP/PV reliability of 0.808 (analogous to Cronbach alpha), meaning that to a satisfactory extent the instrument is measuring the same construct. The person separation reliability of 0.979, indicating that the instrument does an excellent job in ranking the students in the sample according to the construct.

A two-tailed, two sample t-test shows that there was no statistically significant performance difference between the two groups who took the two forms of test, that is, text-based and visual programming language (t(119)= -0.556, p=0.579; Table 3). Further t-tests for each item set also revealed that students performed similarly on all the item sets including 5 and 6 (note only sets 5 and 6 have form variation). These results showed that students performance on this instrument was not influenced by the drag-drop and text-based coding environments. This was expected as the majority of the students in our study had little experience in programming prior to our curriculum and they were not sensitive to the coding environments.

Table 3
Differences of estimated student abilities between the two forms (text-based and drag-drop) based on the whole instrument as well as each item set

	Mean Difference	Std. Error Difference	t	df	Sig.
Overall	-0.097	0.174	-0.556	119	0.579
Item Set 1	0.087	0.061	1.422	119	0.158
Item Set 2	0.065	0.051	1.275	119	0.205
Item Set 3	-0.082	0.140	-0.584	119	0.560
Item Set 4	-0.117	0.069	-1.692	119	0.093
Item Set 5	0.047	0.131	0.362	119	0.718
Item Set 6	-0.136	0.077	-1.775	119	0.078

One-way ANOVA shows that the different classes did perform differently (F (120)=10.79, Sig.=0.000). Further analyses showed that Class II, which had more high performing students, outperformed all other classes. It also showed that Class I was among the lowest group. This suggests that students' CT is possibly related to their general academic ability.

3.1.2 Challenging Items

While examining specific student responses to individual items in pretest, we found the following themes. Students performed poorly on data processing (percentages correct are 4.1%, 14.1%, and 19.0%, for the data category of items 2.3, 3.3, and 6.5).[2] They may have either overlooked or simply failed to integrate data that are otherwise essential to solve the problem. For instance, Fig. 6 shows a student's response to item 6.5, which involves programming the robot to move between squares. This response shows that while the student both figured out the correct algorithm and applied the given syntax to solve the task, he or she did not apply a correct number of steps (2 steps rather than 1, as given in the problem statement) in order to command the robot to move to the next square.

==insert Fig. 6 about here==

Also, students showed weaknesses in terms of representation (the percentages correct on the representation category of items 2.3, 5.5, and 6.4 were 23.1%, 33.9%, and 28.1% respectively). Item 2.3 showed that students had a problem in using symbols to represent variables. Poor performance on item 5.5 and 6.4 revealed that students had difficulty both using representations to express their solutions and interpreting given representations.

Students performed poorly on several items asking them to come up with correct algorithms as well (the percentages correct are 25.62%, 21.49%, 9.9%, 9.1%, and 14.0% on items 4.2, 4.3, 5.5, 6.3, and 6.5). Among those items, low score on 4.2 and 4.3 might be due to students' inability in identifying parallel execution structure (hence coming up with the correct algorithm). Their even poorer performance on 5.5, 6.3, and 6.5 showed difficulties with the necessary algorithmic thinking skill in organizing and ordering problem solving steps correctly.

Although there were a number of students who were not attending to given syntax, one interesting observation drew our attention: Violating given syntax could actually represent good CT, especially in coming up with creative and even more efficient solution. For instance, item 5.5 (see Fig. 3), one of the most challenging items, required students to program the robotic arm to draw a shape most similar to a triangle, while the given commands involved only vertical and horizontal movements. While creating this item, we hoped to see the correct solution as a staircase shape with the expectation that some of them would even use the "repeat" command to improve the efficiency of their code. To our surprise, several students came up with a creative solution that broke the given syntax. Fig. 7 shows that a student managed to draw two slanted sides of a triangle by executing the two movement commands simultaneously (i.e., move vertically and horizontally at the same time), which actually could work. Although our current framework does not accommodate creativity, we recognize the algorithmic and efficient nature of this solution and gave it a full score on those metrics, but deducted a point on the syntax component because the solution does not fully comply with given syntax (please refer to Figs. 10 and 12 to see sample responses with correct syntax for this problem).

== insert Fig. 7 about here==

3.2 Pre/Post Test Comparison

In this section we report results from the pre/post comparison in two classes. Given their

[2] For open-ended items, percentage correct refers to the percentage for reaching level 2.

different performances in the pretest (t(34.98)=-6.258, p<0.0001, d=-1.98), we report their pre/post comparisons separately.

3.2.1 Student Gains

3.2.1.1 Class I

In pretest, Class I was among the lowest performing classes. Table 4 lists a series of paired t-test results. It shows that overall, student performance improvement was statistically significant (p=0.002) and the effect size was moderate (d=0.67). Examining the two contexts of computational thinking application, we observed student gains more so in the robotics programming context (d=0.76) than in the everyday reasoning context (d=0.33). Comparing student gains on the two forms, we did not see statistical difference (p=0.217).

We also compared the gains in total scores between those who started with lower scores in pretest (with scores ranging from 8 to 16; n=10) with those who started with higher scores in pretest (scores ranging from 19 to 32; n=12). There was no statically significant difference between the two groups (p=0.692). The result also holds when only looking at scores in item sets 5 & 6 (p=0.849). This suggests that participation in the robotics curriculum benefited the students who started high as well as those who started low in Class I, on average, to the same degree as observed in this instrument.

3.2.1.2 Class II

In pretest, Class II outperformed all other classes. For Class II (Table 4), paired t-test shows that overall, student performance improvement was statistically significant (p=0.0008) and the effect size was large (d=0.97). Just as Class I, student gains were greater in the robotics programming context (d=0.73) than in the everyday reasoning context (d=0.69). No statistical difference was found for student gains on the two forms p=0.755).

By comparing the gains in total scores between those who started with lower scores in pretest (with scores below or equal to 31; n=8) with those who started with higher scores in pretest (scores above 31; n=7), we saw no statically significant difference between the two groups (p=0.198). The result also holds when only looking at scores in item sets 5 & 6 (p=0.518). This suggests that similar to Class I students, those from Class II with varied pretest scores gain to the same degree on average.

Table 4
Multiple paired t-test results of the two focus classes

Tests	Class I			Class II		
	t	p	Effect size (d)	t	p	Effect size (d)
Overall gain	-3.140	0.002*	0.67	-3.868	0.0008*	0.97
Gain on robotics programming (item sets 5&6)	-3.562	0.001*	0.76	-2.901	0.0005*	0.73
Gain on everyday reasoning (item sets 1-4)	-1.560	0.067	0.33	-2.746	0.008*	0.69
Form A vs Form B	1.276	0.217	--	0.336	0.755	--
Lower vs higher score group (overall)	-0.403	0.692	--	-1.353	0.198	--
Lower vs higher score group	-0.194	0.849	--	-0.664	0.518	--

(item sets 5&6)						

* Numbers with a "*" refer to results that are statistically significant at an alpha level of 0.05 or higher.

In the following, we show several examples to illustrate areas where students improved in light of our CT framework.

3.2.2 Examples of Student Improvement or Lack thereof in Robotics Programming

Many students improved the ways in which they formulate a solution using given syntax. Fig. 8 shows a student's response to the same problem (item 6.3, see Fig. 2) in the pretest (top) and the posttest (bottom). This student used natural language to describe his or her algorithm in the pretest as opposed to given commands to formulate his or her solution in the posttest. This kind of improvement was also common in the text-based coding environment papers.

== insert Fig. 8 about here ==

We also witnessed a few cases where students simply copied the commands they learned from the robotics programming environment, while ignoring the given commands in the test items. For instance, a student used a box "sit down" that was not provided in the item, but had been used in the curriculum. This is both negative and positive since for one thing, students who did so began to use syntactically meaningful representations to solve the problem, but for another, they were incorrectly applying what they had learned from the curriculum to a new context that uses a different syntax.

In general, students in both classes did not improve in terms of data processing. Fig. 9 shows one of the rare cases where a student changed from pretest (top) to posttest (bottom), developing the sense of attending to the quantitative information given in the problem by including the specific steps that the robot needed to walk to accomplish the task.

==insert Fig. 9 about here==

Students did show much gain in representation and algorithmic thinking. In the pretest, many students, possibly daunted by their lack of prior programming experience, either simply left some open-ended questions blank or wrote irrelevant information. Many of these students showed obvious gains in scores on the same items in the posttest. Fig. 10 showed a student's correct solution of item 5.5 in the posttest, and the same student did not say anything in his or her pretest. Given the nature of blank or irrelevant answers in the pretest, it is difficult to infer from pre/posttest comparison that a student improved in terms of which CT component. However, repeated observations of this common pattern suggest that students who performed better in either organizing or representing algorithms.

==insert Fig. 10 about here==

As discussed previously in section 3.1.2, we know that the students might have difficulty in understanding parallel execution. Fig. 11 shows the improvement of a student's response from pretest (top) to posttest (bottom) to item 6.3 (see Fig. 2 for the first half of the item). This difference in how they answered the question suggests a better understanding of executing the

turning action and the say action in a parallel processing structure rather than executing them in a serial manner (the posttest response also showed improvement in the syntax aspect, including the added "start" and "end" commands and the commands being wrapped in boxes).

==insert Fig. 11 about here==

Fig. 12 shows another student's pretest (top) and posttest (bottom) comparison in solving item 5.5. The comparison showed clearly improvement in coding efficiency as in the posttest the student used the command "repeat" to avoid redundancy.

== insert Fig. 12 about here ==

3.2.3 Student Improvement in Everyday Reasoning

Apart from significant gains in the coding context (item sets 5-6), the students from the two classes gained differently in the everyday reasoning context (item sets 1-4; see Table 4): The students from Class II gained about the same degree in the everyday reasoning context as in the programming context whereas those from Class I only gained a moderate degree in the everyday reasoning context.

By mapping our CT framework on the responses as shown in table 5, we could see that students gained differently on different dimensions (the syntax component was not present since syntax was simply not applied in everyday scenarios). To be specific, both classes gained significantly in terms of algorithm. Class I had a statistically significant gain on efficiency and Class II on data.

Table 5

Multiple paired t-test results of two classes in terms everyday reasoning

Tests	Class I			Class II		
	t	p	Effect size (d)	t	p	Effect size (d)
Gain on Data	-0.093	0.464	0.02	-2.988	0.005*	0.75
Gain on Representation	-0.295	0.386	0.07	0.565	0.710	0.14
Gain on Algorithm	-2.156	0.022*	0.48	-2.766	0.007*	0.69
Gain on Efficiency	-1.789	0.045*	0.40	-1.651	0.060	0.41

* Numbers with a "*" refer to results that are statistically significant at an alpha level of 0.05 or higher.

Fig. 13 shows a student's improvement in coming up with a feasible algorithm in solving an everyday reasoning problem in the posttest. The same student overloaded one washing machine in the pretest, which resulted in a wrong working flow of solving the problem. The right side of the equations given in both pre/posttest in Fig. 14 shows another student's realization of using a letter 'n' to represent the variable nature of the answer in the posttest as opposed to writing a specific number in the pretest. This is important since it requires a certain level of abstraction skills.

== insert Fig. 13 about here ==

== insert Fig. 14 about here ==

3.2.4 Summary and implication

Our analyses showed that the instrument we developed had good psychometric properties. Overall, the instrument was consistent in measuring one core construct and it was able to differentiate the fifth grade students' abilities, presumably, in CT. The results revealed that students performed similarly on the two forms (drag-drop versus text-based) in pretest. Our pre/post comparison of two classes showed that the robotics curriculum did help improve students' CT, especially in the context of robotics programming. We also presented examples of student improvement in light of our CT framework. Furthermore, the results showed that the curriculum helped those who started low as much as those who started high, and the gains were not dependent on the forms of test they took. Plus, strong evidence showed that students as young as 5th graders could manage to grasp CT to some degree, and mixed results revealed some promise for young children using CT to reason everyday scenarios. Yet, our results suggest the need to strength our robotics curriculum to better facilitate (and also revise and refine our instrument to better evaluate) students' development of CT, especially in the following aspects.

Across all CT components, our students, even in their posttest, had difficulties following a given syntax. One possible explanation is that in the visual programming environment used in this curriculum, the syntax aspect is not as obvious as it is in other more syntactically strict languages (Java, C, etc.). Given the importance of syntax in programming, this result highlights the need for further emphasis on correct syntax. We also noticed that in the pre/posttest, some students came up with creative and unexpected algorithms by sacrificing the correctness of syntax. We recognize that such solutions could work for programming a real robot (and even more efficient than the other way around). Additionally, our assessment does fortunately accommodate such possibilities. Recognition of these algorithms raises a subsequent question, how to code creative answers? Such creative answers can conflict with other established components of CT, yet they can be beneficial and ideally encouraged in the curriculum.

Data processing is an important aspect of CT, yet it is difficult to isolate. Our current items targeting the data component are either intertwined with other aspects in multiple-choice questions or assessing students' sensitivity to data at a surface level. In our future version we need to come up with items that measure data processing skills at a deep level. Based on our observation, many students could not even successfully deal with data on its superficial level. This may reflect a weakness in our curriculum, which did not explicitly address data at the conceptual level.

Algorithmic thinking is a core ingredient of CT, and was difficult for our students. According to our pretest data, the fifth graders usually got stuck on items that are related to algorithm, especially on those involving parallel execution. Although we observed student gains in those items in the posttest, it is still unclear that to what extent did our curriculum factor into the achieved improvement due to the relatively small sample size.

Finally, it is important to improve the connections between the robotics programming environment and everyday reasoning in our curriculum. Based on the pre/posttest comparison, students overall did not gain as much in dealing with CT applied to everyday life as that to programming robotics. Transfer is knowingly difficult (Bransford & Schwartz, 1999). However, by looking at student performance at the class level, one class (Class II) improved significantly, while the other (Class I) only moderately. Also, both classes improved on different dimensions (Class I improved on algorithm and efficiency, whereas Class II on data and algorithm with

different significance levels). This shows that transfer is a complex process that is possibly related to multiple factors such as student variations and differences in instruction. Our next steps could incorporate more coding examples connected to everyday reasoning and ask students to reframe everyday tasks to coding problems.

4. Limitation

Our work has several limitations. First, we constructed our instrument with an emphasis on robotics programming. Others who wish to use or develop a similar instrument need to consider their own programming context. Also, it was not entirely clear to us how the individual lessons within our curriculum helped improve (or not) specific aspects of CT. Future work will draw more detailed connections between the CT dimensions and particular lessons. A related question is whether students are learning CT in their regular fifth grade instruction and the potential issue of test/retest effect, and a control group in future implementations may address these concerns. Furthermore, additional issues raised by the current structure and content of our assessment may possibly influence the quality of students' responses. For example, there may have been linguistic and reading challenges associated with the assessment. Another factor is test fatigue as it took some students up to an hour to complete it. Finally, we acknowledge that we had a small sample size (especially, we only had two focus classes for the pre/post comparison) and all of the students were drawn from one school site.

5. Conclusion

Taking steps towards developing a valid and reliable instrument to measure elementary students' CT is challenging. First, there is a lack of consensus in the field in terms of CT definition. Many versions of CT definitions are vague at the best. This poses a significant barrier to the operationalization of CT in concrete assessment items. Second, many elementary students have limited programming experience. This creates a need for an instrument that can be administered as pre/posttest and can apply across different programming platforms. Our work, pilot in its scope, can contribute to the field in several ways. We developed an instrument on CT based on operationalizable components of a CT framework. We developed specific coding rubrics for each item based on each component of the framework. Also, our instrument used two contexts (robotics programming and everyday reasoning) to assess students' CT application, which sheds light on the near and far transfer aspect of CT. This is critically important as CT is a valuable skill for all learners, may apply to daily problem-solving activities, and may apply to many other STEM learning areas.

Acknowledgement

[to be inserted once accepted]

Reference

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7), 832-835.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *Acm Inroads*, *2*(1), 48-54.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers &*

Education, *72*, 145-157.

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. *Review of research in education*, *24*, 61-100.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). How people learn.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).

CSTA (2011). Operational Definition of Computational Thinking for K–12 Education, 2011. Accessed from http://www.csta.acm.org/Curriculum/sub/CompThinking.html

Cuny, J., Snyder, L., & Wing, J. (2010). Demystifying Computational Thinking for Non-Computer Scientists, work in progress.

Dagienė, V., & Stupuriene, G. (2016). Bebras-a Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education-An International Journal*, (Vol15_1), 25-44.

Duncan, C., & Bell, T. (2015, November). A Pilot Computer Science and Programming Course for Primary School Students. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 39-48). ACM.

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

Haskell, R. E. (2000). *Transfer of learning: Cognition, instruction, and reasoning*. Academic Press.

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010, September). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 59-66). IEEE.

Lewis, C. M. (2010, March). How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 346-350). ACM.

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

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2010). *Stuck in the shallow end: Education, race, and computing*. MIT Press.

National Research Council (US). (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking*. National Academies Press.

Seiter, L., & Foreman, B. (2013, August). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 59-66). ACM.

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

Tew, A. E., & Guzdial, M. (2011, March). The FCS1: a language independent assessment of CS1 knowledge. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 111-116). ACM.

Wainer, H., & Wang, X. (2000). Using a new statistical model for testlets to score TOEFL. *Journal of Educational Measurement*, *37*(3), 203-220.

Wang, W. C., & Wilson, M. (2005). The Rasch Testlet Model. *Applied Psychological Measurement*.

Weintrop, D., Beheshti, E., Horn, M. S., Orton, K., Trouille, L., Jona, K., & Wilensky, U. (2014, July). Interactive Assessment Tools for Computational Thinking in High School STEM Classrooms. In *International Conference on Intelligent Technologies for Interactive Entertainment* (pp. 22-25). Springer International Publishing.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 215-220). ACM.

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

Wu, M. L., Adams, R. J., & Wilson, M. R. (1998). *ACER ConQuest: Generalised item response modelling software.*

Yardi, S., & Bruckman, A. (2007, September). What is computing?: bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the third international workshop on Computing education research* (pp. 39-50). ACM.

Figure 1. Sample item in the everyday scenario.

1.2 (Information from the stem of the item set: You are running 4 loads of laundry in a fast washing machine and each load of clothes takes 10 minutes to run) If you have 2 washing machines that can work simultaneously (i.e., at the same time), how long will it take to run all the loads?

A. 10 minutes
B. 20 minutes
C. 30 minutes
D. 40 minutes

Figure 2. Two forms of robotics-programming commands (bottom left: text-based; bottom right: drag-drop) in the same problem (top); The rest of the problem prompts the students to write a code sequence so that the robot walks a certain path (see Fig. 11).

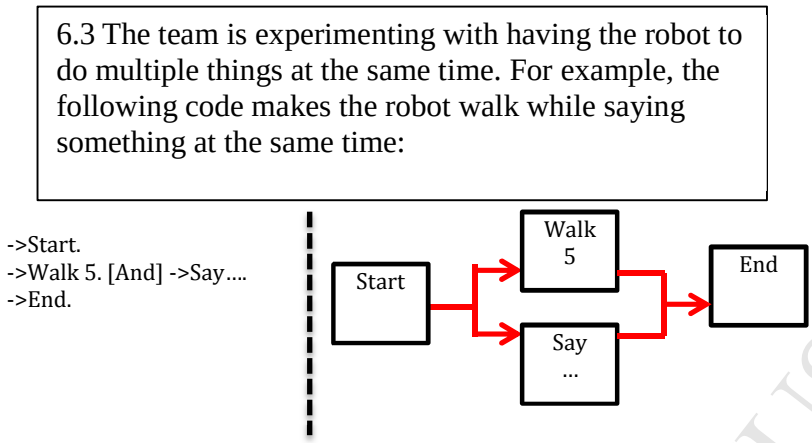
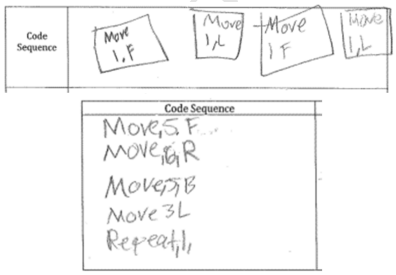
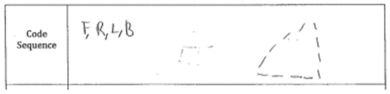
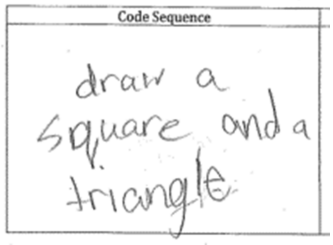
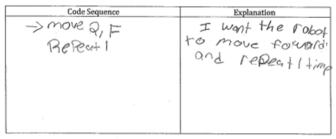
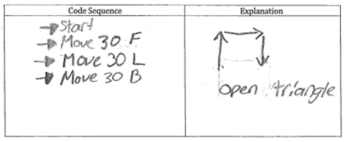
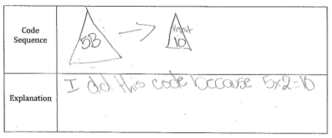


Figure 3. Rubric sample for CT components of item 5.5, which asks students to generate and explain a code sequence to have a robotic arm to draw a shape most similar to a triangle using the given commands that can only draw straight lines.

Component: Expected performance	Level	Description	Sample response
Syntax: Answer is written in codes instead of natural languages.	2	Commands in the solution obey the given syntax totally	
	1	Commands in the solution obey the given syntax partially	
	0	Commands in the solution are in natural language or do not obey the given syntax	
Representation: Show ability to manipulate given commands to represent the solution and provide correct explanation.	2	Commands in the solution match the explanation exactly (student drawings on the paper should also be counted as explanation)	
	1	Commands in the solution match the explanation partially (student drawings on the paper should also be counted as explanation)	
	0	Commands in the solution do not match the explanation (student drawings on the paper should also be counted as explanation)	

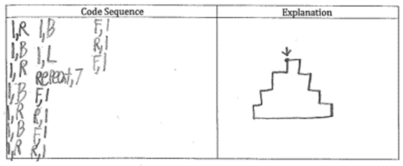
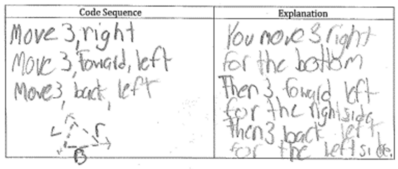
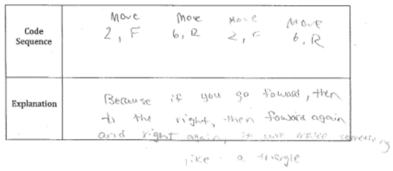
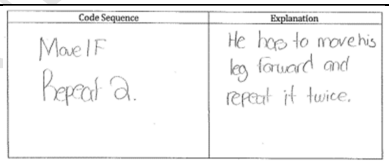
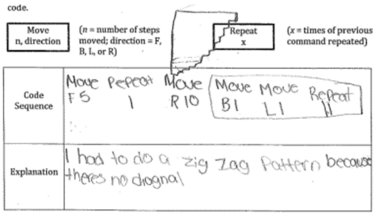
Algorithm: The coded steps solve the problem.	2	Solution algorithm solves the problem	 
	1	Solution algorithm solves the problem partially	
	0	Solution algorithm doesn't solve the problem	
Efficiency: Solution is successful with minimum complexity.	2	Problem solved without redundant codes	
	0	Problem solved with redundant codes	

Figure 4. The iterative process of coding open-ended items.

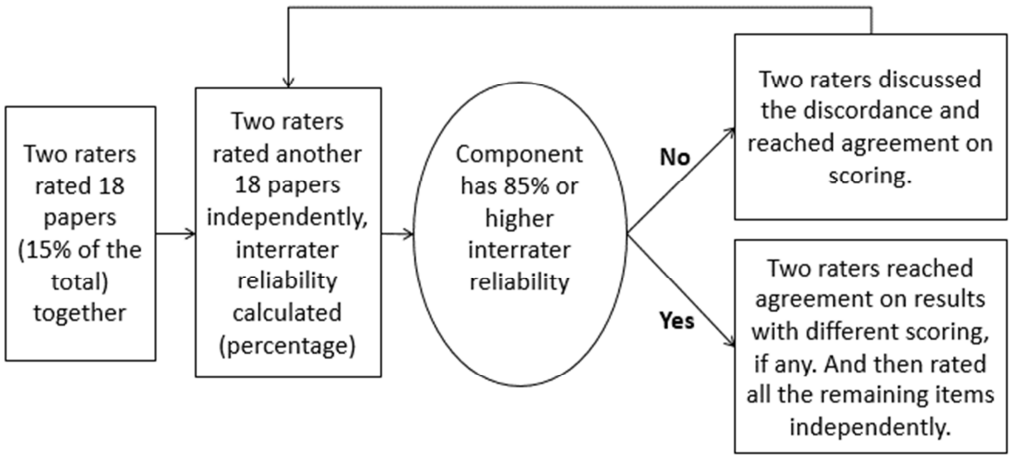


Figure 5. Map of latent distributions and response model parameter estimates: The Y-axis is in logit scale; each ‘x’ represent ‘1.4’ students; the first column represents the distribution of student abilities based on the whole instrument; the next six columns represent the distribution of student abilities based on each test set; the last column indicates the distribution of the difficulty levels of each item.

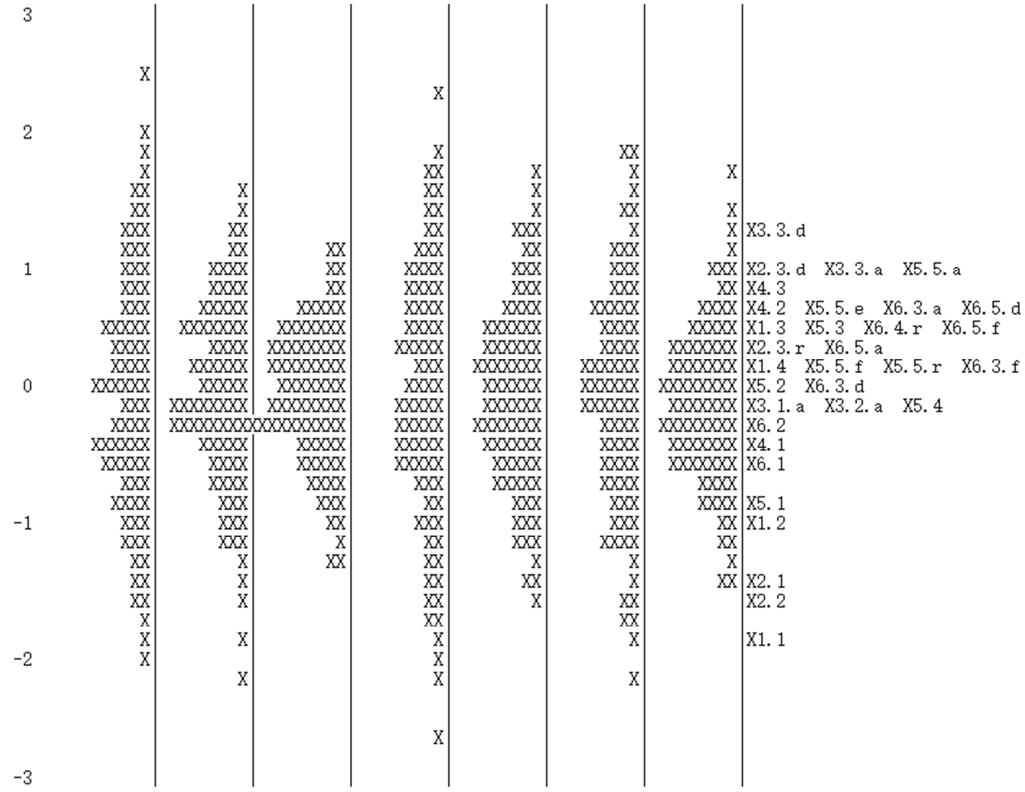


Figure 6. Sample student answer that did not integrate data correctly

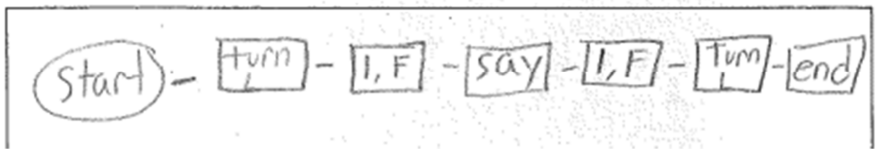


Figure 7. One example of creative solution to item 5.5

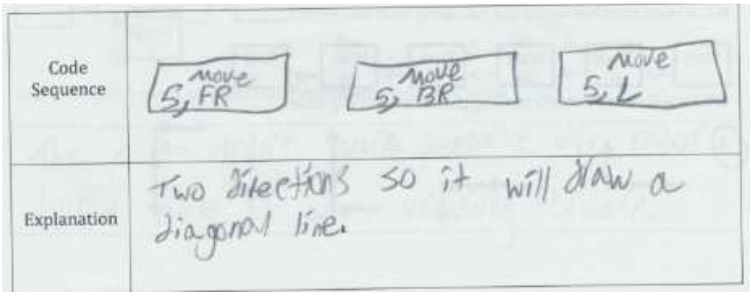


Figure 8. Improvement in formulation of solution to item 6. 3 using given syntax from pretest (top) to posttest (bottom) from the same student.

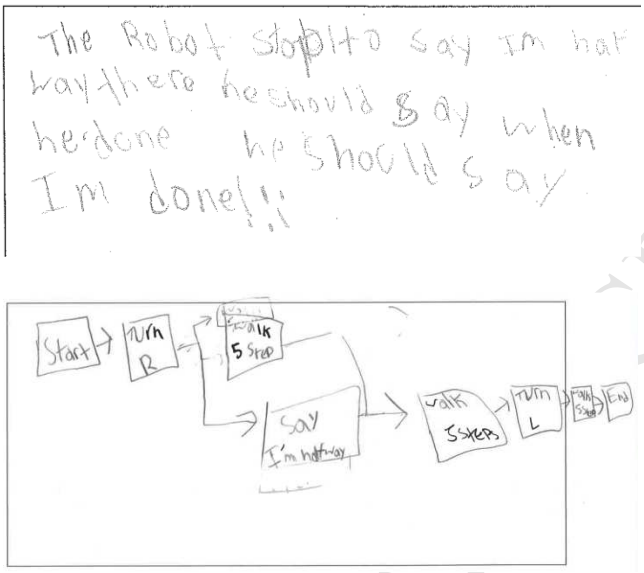


Figure 9. Improvement in data processing from pretest (top) to posttest (bottom) for the same student.

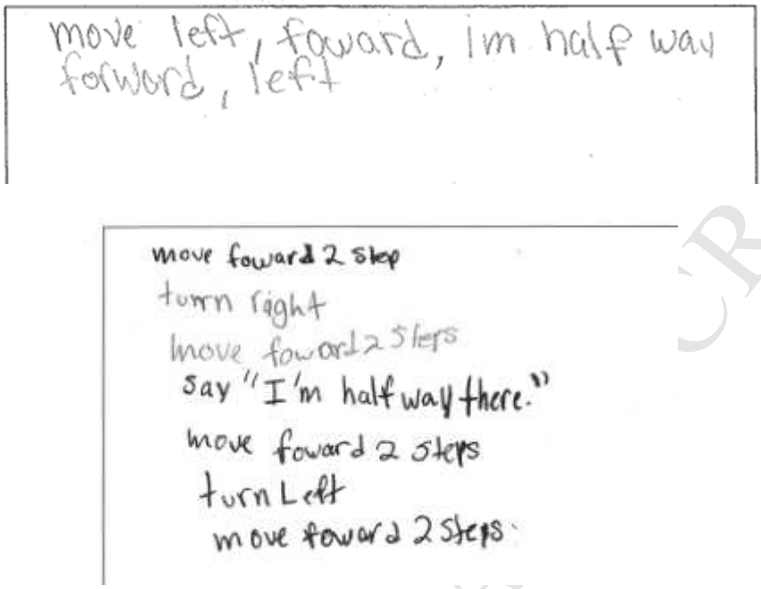


Figure 10. Student who left item 5.5 blank in the pretest answered the same item correctly in the posttest

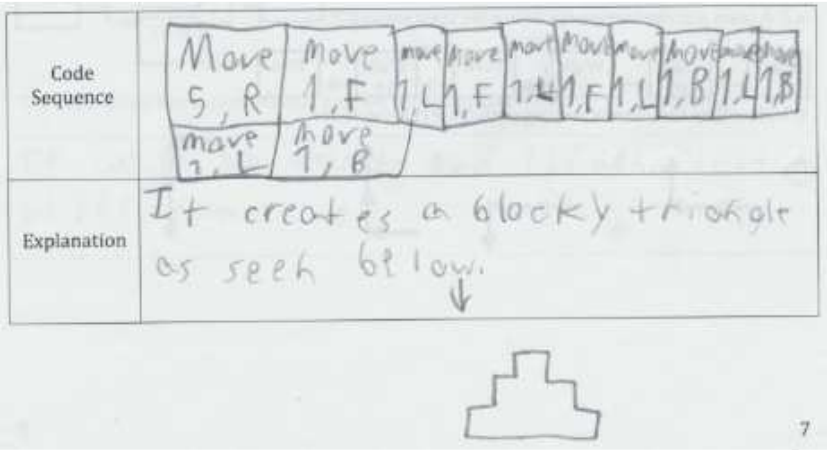


Figure 11. Improvement in applying simultaneous execution algorithm to item 6.3 from pretest (top) to posttest (bottom) from the same student.

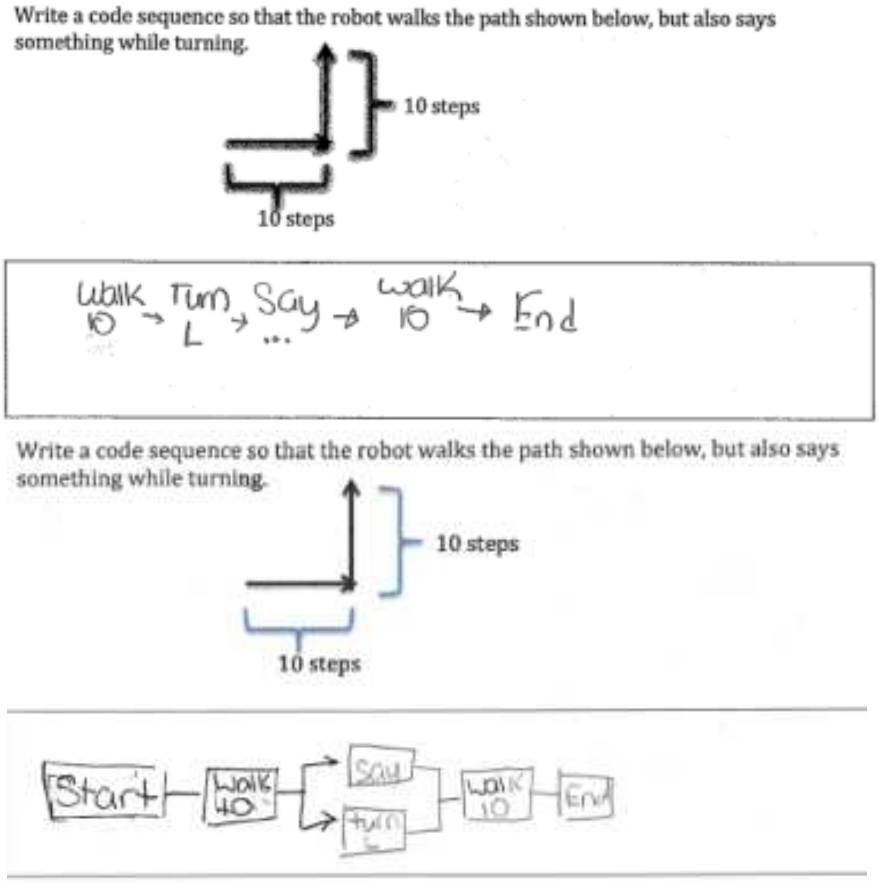


Figure 12. Improvement in efficiency of code from pretest (top) to posttest (bottom) from the same student.

Code Sequence	Explanation
1. Move 5 B. 2. Move 5, R. 3. Move F. 1. 4. Move 1 P 5. Repeat 4, step 3 and 4	

Figure 13. Improvement in algorithm in an everyday scenario from pretest (top) to posttest (bottom) from the same student.

3.1. If the dark and light clothes cannot be mixed and your machine holds one pile of laundry at a time. Also, you want to use the fewest rounds possible. The table below is a loading plan your sibling came up with. Does this plan meet all the conditions? If yes, go to the next question; if not, write down your loading plan using the blank row.

Round	1	2	3	4	5	6
Your sibling's plan	P1	P2	P3, P4	P5	P6	
Your plan	P1, P2, P3	P5				

3.2. What if you have 2 washing machines that can run simultaneously. All other conditions still apply: the dark and light clothes cannot be mixed; each machine holds one pile of laundry at a time; use the fewest rounds possible. Fill out the loading plan table below.

Round	1	2	3	4	5	6
Machine 1	P4, P6					
Machine 2	P1, P2, P3	P5				

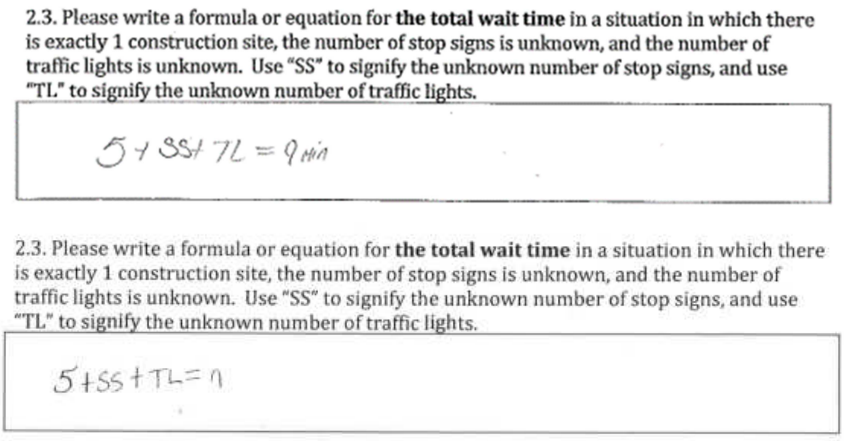
3.1. If the dark and light clothes cannot be mixed and your machine holds one pile of laundry at a time. Also, you want to use the fewest rounds possible. The table below is a loading plan your sibling came up with. Does this plan meet all the conditions? If yes, go to the next question; if not, write down your loading plan using the blank row.

Round	1	2	3	4	5	6
Your sibling's plan	P1	P2	P3, P4	P5	P6	
Your plan	P4, P6	P2, P3	P1	P5		

3.2. What if you have 2 washing machines that can run simultaneously. All other conditions still apply: the dark and light clothes cannot be mixed; each machine holds one pile of laundry at a time; use the fewest rounds possible. Fill out the loading plan table below.

Round	1	2	3	4	5	6
Machine 1	P4, P6	P5				
Machine 2	P2, P3	P1				

Figure 14. Improvement in representation in an everyday scenario from pretest (top) to posttest (bottom) from the same student.



Assessing Elementary Students' Computational Thinking in Everyday Reasoning and Robotics Programming

Research Highlights

- Assess computational thinking (CT) in ways that are independent of platform
- Assess fifth graders' CT in both everyday and programming settings
- Psychometric analysis of the instrument shows high quality