Kashihara, A. et al. (Eds.) (2024). Proceedings of the 32nd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Who Is a *Good* Computational Thinker? Mapping Behavioral Dispositions of Middle-School Children Based on Real-Life, Algorithmic Tasks

Shashaank V. PINNAMARAJU*, Lazar TONY & Anveshna SRIVASTAVA^
Department of Cognitive Science, IIT Kanpur, India
*shashaank23@iitk.ac.in, ^anveshna@iitk.ac.in

Abstract: Computational Thinking (CT) is a critical thinking ability we want children to develop for efficient and effective problem-solving. Aligned with the recent unplugged, non-digital approach to developing CT, we show, through this ongoing study, in a low-resource, no-computer context, how behavioral dispositions of children map onto the established facets of being a 'good' computational thinker. Twenty middle-school children (Mean age 14.6 years, F= 12) volunteered to participate in an experimental study, where they were randomly grouped into one control (n=9) and two experimental groups (n=5, 6). The experimental groups were administered a 'sorting' task and a 'sorting + simulation' task, respectively, and all the groups were administered a standardized CT test (CTt) as the pre and post-test. We use a mixed-method research design to analyze our data. Our quantitative methods revealed no significant difference between the pre and post-CTt scores but better performance on 'arrow' questions than on 'visual-block' questions. Our qualitative methods provide insights into students' problem-solving and learning processes. They reveal their ability to 'persist', 'abstract', 'verify', and render a 'generic' solution mapped onto the established CT skills of iteration, abstraction, debugging, and algorithm generating.

Keywords: Computational thinking, Middle-School, Unplugged activity, Algorithmic thinking, Abstraction, Simulation

1. Introduction

Computational thinking (CT) is a critical 21st-century ability that goes beyond effectively harnessing the computing powers of computers or managing data by efficiently handling technology (Shute, Sun, & Asbell-Clarke, 2017). The theoretical construct of 'CT' emerged from the field of computer sciences, where it was argued that college students should learn the 'theory of computation' and programming skills (Guzdial, 2008). However, with the advent of the first programming language for children- LOGO, the focus was on developing procedural thinking in children and helping them to construct their mental structures (Papert & Harrel, 1991). Wing (2006) took the concept forward and re-introduced CT as a cognitive construct and described it to include a range of mental processes that are involved while formulating a problem, rendering a generic solution, the steps of which could be replicated by another human or a machine (Wing, 2006, 2011). Such concept rendering is based on the idea that children should think computationally and apply principles of computer sciences to solve real-world problems with or without using computers to program (Caeli, & Yadav, 2020).

These 'without computer' or 'unplugged' contexts offer fertile ground for CT principles to flourish and be used as an analytical approach to problem-solving (Caeli, & Yadav, 2020). This is because it removes the physical constraint of working with a computer, offers direct engagement with CT principles, and provides ample opportunity to develop CT temper. It is helpful in low-resource settings where access to computers and the internet is still a distant dream. Unsurprisingly, it has been argued that teaching CT skills without a computer interface

is more effective and efficient for students from non-computer backgrounds (Kim, Kim, & Kim, 2013). Our study is situated in a resource-constrained setting where the participating middle-school students had limited computer access but were theoretically exposed to computer sciences as part of their school curricula.

1.1 This Study

Although there is no consensus on a standard definition of CT, it is generally agreed to be composed of the following skills (Shute et. al., 2017):

a) Decomposition- decomposing a problem into simple manageable parts.
b) Abstraction- identifying inherent patterns/rules/structures and applying that understanding in a different context.
c) Algorithm- designing ordered instructions to solve a problem such that another human or machine could replicate the steps.
d) Debugging- identifying errors and fixing them.
e) Iteration- repeating a set of actions till the ideal solution is reached, and
f) Generalization- transferring of skills for efficient & effective problem-solving.

These skills could be used for analyzing problems across disciplinary domains (Yadav, Hong, & Stephenson, 2017), and, like other analytical skills, our minds could be trained to use CT skills to solve complex world problems. In this study, we had the twin objectives of assessing school students' CT competencies and the efficacy of a simple algorithmic task, alone or in combination with a simulation task. We used a standardized computational thinking test- CTt (Román-González et al., 2017) to measure students' CT competencies. The algorithmic task involved a basic sorting task, and the simulation task involved following certain algorithmic steps to reach a solution.

This study concerns middle schoolers who have had limited exposure to computers. Our specific research questions for this study are-

RQ1 How does the algorithmic sorting task affect students' CT competencies? How does the combination of 'sorting' and 'simulation' tasks affect students' CT competencies?
RQ2 What insights could be drawn from students' problem-solving exercises of the 'sorting' task &/or 'sorting + simulation' task?
RQ3 How does a given algorithmic solution affect students' originally constructed sorting algorithm? How do students interact with this given algorithmic solution?

2. Method

2.1 Research Design

We use a mixed-method research design to answer our research questions (Cresswell and Cresswell, 2017). Our method involved a convergent parallel design, where the quantitative and qualitative methods complemented each other as all data was collected and analysed in a parallel fashion. Quantitative methods were used to analyze students' performance on the CT measure and qualitative methods were used to analyze learning and methodological processes involved in students' problem-solving.

2.2 Participants

20 middle school students (Mean age = 14.63, SD =2.45, Males=8) volunteered to participate in the study. Students' assent and their parents' consent were received for the study. The study was conducted in their regular classroom with the permission of the school authorities. The study was approved by the institute's review board.

2.3 Instrument

We use the Computational Thinking test (CTt) designed to assess middle-school students' ability to solve problems based on fundamental computing concepts and the basic structure of programming languages (González, 2015; González et. al., 2017). The CTt was originally designed for 12-13-year-old Spanish kids and is now widely used worldwide for its increasingly reliable and valid results (Tang et al., 2020). There are 28 items in the CTt measure, constructed to assess competence in basic sequences, loops, conditionals, iterations, and functions. Each question is a multiple-choice question with four given options and only one correct option. It is a pen-paper physical test estimated to take 45 minutes to complete.

The CTt items are presented as a 'maze' or 'canvas' and invite problem-solving using arrows and visual blocks (see Figure 1). The questions increase in complexity as one moves forward in the problem-solving exercise. The CTt items also vary on the kind of response they elicit from the problem-solvers. For instance, questions involve either '**writing**' of steps, **'completion**' of incomplete steps, or '**debugging**' the steps.

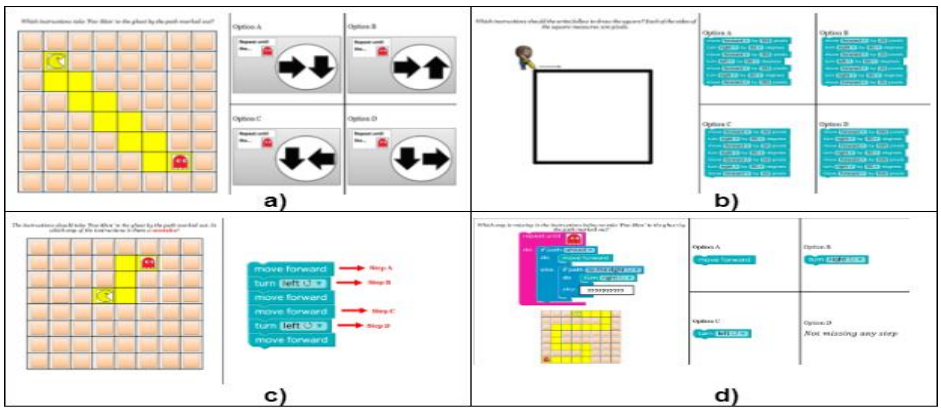

Figure 1. Sample questions from the CTt: a) A 'maze' question with arrows, b) A 'canvas' question with code blocks, c) A debugging 'maze' question with code blocks, and d) A completion 'maze' question with code blocks.

2.4 Procedure

All students were administered the CTt as the pre-test and the post-test. After the pre-test, students were randomly grouped under one 'control' and two experimental groups based on the 'sorting' or 'sorting + simulation' tasks. Table 1 gives the break-up of the groups and the corresponding details about the students. The students were in the age range of 13-16 years.

Table 1. *Break-up of the Groups*

Group	No. of Students	Mean age (yrs.)	S.D.	M/F
Control	9	14.3	0.97	3/6
Sorting Task	5	14.4	1.14	1/4
Sorting Task +Simulation	6	13.83	0.75	4/2

The control group students had no intervention, the 'sorting' task group students were asked to do a physical sorting task and verbalize the corresponding algorithm used for doing it, and the 'sorting + simulation' group students were asked to do the sorting task and simulate a given algorithm to observe and compare with their original algorithm. Figure 2 illustrates the overall flow of the study. The study was spread over one week and the students in the experimental groups were interviewed soon after the intervention, while all groups were interviewed after the conduct of the post-test.

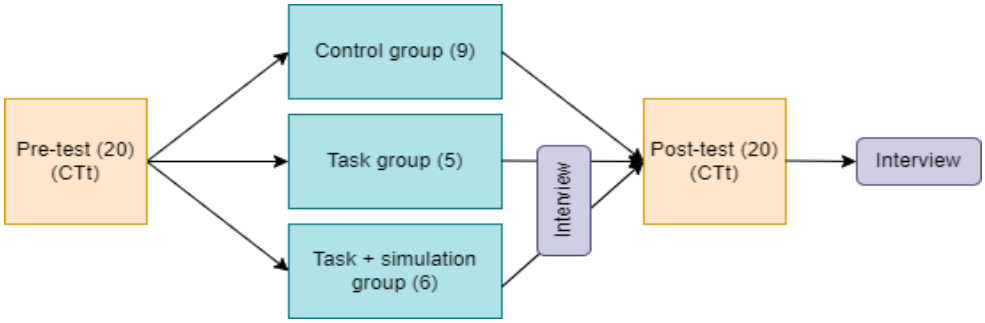

Figure 2: The procedure of the study.

2.5 Intervention

2.5.1 Sorting task and generating an algorithm

The sorting task was done individually. It required students to sort six identical-looking hand-made cylindrical rods with different weights in an ascending order using a balance scale. The balance scale was uncalibrated and made from regular household goods. Participants could weigh individual rods to compare any two of them at a time (Figure 3).

*Figure 3***.** A student attempting the sorting task. The weighing apparatus is visible in the foreground (the student's face is smudged to protect his identity).

After completing the task, students were asked to 'check' if the rods were placed in an ascending order (see Figure 5). On discovery of an error, they were given a chance to correct it. This process was repeated until an error-free sequence was obtained. They were then asked to write down, succinctly yet clearly, all the steps they had followed in solving the 'sorting' task. They were given an added constraint of writing their problem-solving 'algorithm' such that if anyone else were to follow their steps, they would reach the same error-free sorted state. No other constraints were placed on the answer's language, length, or nature.

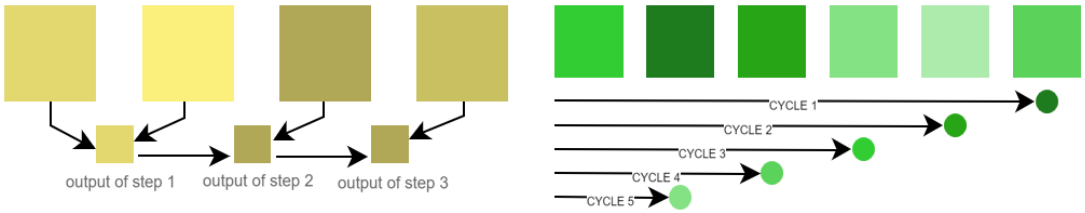

Figure 4. (left) An illustration of the steps constituting a single cycle of sorting. The heavier rod from each comparison has been represented with a smaller stick on the connecting arrow. (right) An illustration of the selection-sort technique. The darker shade represents a heavier weight.

They were then given a short 'incorrect' algorithm using a shorthand notation (see Figure 8). This shorthand notation used simpler terms like the position of the slots (S1, S2,

etc.), ‘H’ for ‘heavier’ weight, and ‘L’ for ‘lighter’ weight. This exercise aimed to expose them to the shorthand notation and give them some idea of what an algorithm might look like. Once the incorrect algorithm was simulated, they were asked to generate a sorting algorithm. After they had generated their ‘sorting algorithm’, they were asked to follow their steps to verify the generality of their algorithm. Finally, they were interviewed to assess their understanding and perception of the task.

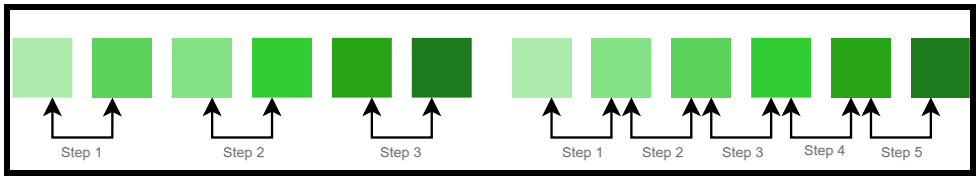

Figure 5. The incorrect method of verifying whether the rods are in ascending order(left) and the correct verification method (right)

2.5.2 Simulation task

A day after the sorting and the algorithm-generation task, students were given a correct algorithm. This algorithm was modelled after the standard selection sort technique, which utilizes the methodology of making comparisons until we find the heaviest rod kept at the end of the sorting set (Figure 4, illustration on the right).

This process is repeated with the remaining rods until all rods are correctly sorted in ascending order. The written algorithm used the same shorthand notation that the students were given in the earlier ‘sorting’ task phase. They were then asked to simulate (follow the steps) and check if it was a valid solution to the problem. They were then further asked to compare their initial algorithm with the given algorithm and again given a chance to revise their algorithm if desired. An interview was then conducted to understand their procedural understanding and their thoughts on the new algorithm.

2.5.3 Interviews

In addition to the interviews after the ‘sorting’ and the ‘simulation’ tasks, students were also interviewed after their post-test. They were asked about their preference for specific questions in the CTt, their overall perception of the test, their difficulties and confidence in tackling the questions, the relevance of the tasks, etc.

2.6 Data sources

Students’ written responses on the CTt, the video recordings of the ‘sorting’ task and the ‘task + simulation’ process, and the video transcript of their interviews were the major data sources for the study. We also assessed certain quantifiable and behavioral parameters. Quantifiable parameters included the time the participant took to sort, the number of attempts they took to sort, the length of the free-form steps, etc. Behavioral parameters included qualitative parameters like the level of abstraction used for the free-form steps, reported confidence, the generality of the algorithm, etc.

3. Analysis & Findings

3.1 Quantitative analysis & findings (RQ1)

Students’ performance on the CTt showed no significant difference between the pre and the post-test for all three groups. It is to be noted that the mean pre-test scores for the three groups were not significantly different from each other, i.e., the three groups were not inherently different in computational competencies.

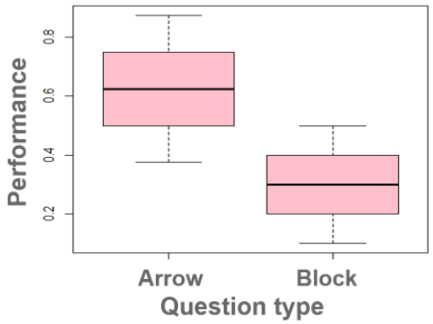

Figure 6. Box plot representing students' performance on 'arrow-based' questions (X-axis) compared with 'visual-block' questions.

A paired t-test was performed to compare the pretest and the post-test scores. There was no significant difference in pre and post-test scores; t(19) = 9.11, p = .37. Group-wise tests were conducted and the difference between the test groups and the control group was also measured. All tests returned nonsignificant p-values.

Students performed significantly better on 'arrow-based questions than on 'visual-block' questions. A two-sample t-test was performed to compare category-wise score ratios (proportion of questions correctly answered). A significant difference was found between the arrow and block-based questions; t(19) = 9.12, p < .001 (Figure 6).

Students' performance on the CTt pre-test was moderately correlated with their academic performance in English language and Computer Science. We found the highest correlation of the CTt scores on the pre-test with students' academic scores on Computer science (r= 0.448), followed by English (r=0.34), Math (r=0.27), & Science (r=0.16).

3.2 Qualitative analysis & findings

To understand the processes involved and the methodological approaches taken to solve the 'sorting' and the 'sorting + simulation' task, we present case studies of four students, two belonging to each group. The case studies describe students' academic performance, approach, process, and metacognition.

3.2.1 Case Study of Students A & B

3.2.1.1 Performance on the CTt measure and general academics.

Student A's academic performance was below average; she was ranked 15th in a class of 20. This was opposed to her performance on the CTt. Her CTt score was the highest among the girls, and she stood fourth in the sample with a score of 14, whereas the average score was 11.2. Student B's academic performance was average; he ranked 11th in a class of 20. His performance on the CTt was below average, scoring 10.

3.2.1.2 Approach

Student A had an inquisitive and confident demeanor. As soon as she reached the set-up, she noticed the microphone and curiously asked- *"What is this (mic) for?"*

The first author explained the purpose of the microphone, and then she was instructed about the task's details. On the other hand, Student B was slow and methodical in his approach but was confident. He did not look for affirmation once he started doing the task. He made sure he understood the instructions before he began. Student A started working on the

given task as soon as the briefing was finished. Unlike most other kids, she started weighing the rods from the leftmost end and continued sequentially.

3.2.1.3 Process

Student A seemed to use a 'fixed' method to go about the sorting task because she kept repeating this process until she was satisfied with the order, i.e., she kept returning to the first rod at the end of every cycle of weighing consecutive rods. This approach is similar to the ideal approach suggested in Figure 4. Student B took the longest to sort, as he did the task cautiously. He verified each step twice to make sure the rods were correctly weighed.

Student A understood the logic of verifying the order in one go and did not make errors. She weighed consecutive rods and checked if the second rod was heavier than the first in each case. Student B did not require any help with the correct verification algorithm.

3.2.1.4 Metacognition & Behavioral Attitude

Student A expressed disinterest in writing after recognizing that the algorithm would have many steps. *"I'll have to write a lot",* she admitted. It must be pointed out that this cognition itself is of value. Most students started writing some steps after a little thought, but she could simulate writing down the steps for the entire process and realized it would take much time and effort. This reflects her ability to simulate tasks before performing them.

When she realized her algorithm wasn't working, she used both methods, starting over (creating an algorithm from scratch) and modifying (changing her algorithm, i.e. debugging), showing that she was 'flexible' with her approach, signifying that she understood the two procedures (Detterman, & Sternberg, 1993). She was also eager to find a solution. She was taken aback when her first attempt failed and initiated the second attempt. During the interview, she said she could sort any rod arrangement. When asked if she could write a generic algorithm for sorting, she replied that she could do that if she could repeatedly verify her steps and correct them to accommodate any error in the arrangement. However, students A & B wondered at their inability to write an algorithm to sort a randomly shuffled bunch of rods.

3.2.2 Case Study of Students C & D

3.2.2.1 Performance on the CTt measure and general academics

Student C was the third-best academic performer (90th percentile) in class. Conflictingly, she had the lowest score on CTt. Her performance on language examinations was good. Student D had the lowest academic score in the class. Her score of 9 was below average.

3.2.2.2 Approach

Student C was diffident and required validation after each move. The experimenter repeatedly reminded her that there was no 'the' right method; she could perform the task as she wanted. Student D also displayed a similar apprehension. She wasn't sure how to go about things and appeared to weigh rods randomly when requested to begin the task.

3.2.2.3 Process

After repeated failures, student C reformulated her strategy to achieve the desired outcome by adopting invalid means and using intuitive weight estimates. Instructions were given not to do so, but she persisted. She was informed that her process of estimating weights using her intuitive sense was error-prone and that she could use the balance scale, but in vain. Student D also resorted to a similar strategy. Once she started weighing the rods, she realized she could intuitively assess their weights by simply picking them up. Even when asked to write

steps in her own words, student C suggested using her intuitive sense of weights to place the rods in order. She was also unable to use the shorthand notation introduced by the experimenter. Strangely, she kept affirming that she understood what to do. After multiple trials, she managed to write a little but kept struggling. Student D was able to write an apt sorting algorithm. Her steps were not explicit enough to be simulated by another being/machine but gave an abstract description of the required process.

3.2.2.4 Metacognition

Student C seemed to exhibit dissonant behavior. She shared that the most interesting part of the task was weighing the rods on the scale but did not use it for the task itself. The balance was also not used during the writing of her free-form algorithm. Both C & D claimed that they understood their mistakes and could solve the problem with more time. Yet their answers did not reflect that. Their solution was failing because it was written for a specific initial order. They thought the problem was due to some minor error in their steps.

3.2.3 Interaction with the given algorithm (RQ3)

The simulation algorithm provided by the experimenter involved a selection sort-like algorithm (Figure 4). This process was iterative. Upon trying the simulation algorithm out, three of the five students noticed the iterative nature of the process. A student concluded that any set of steps would produce a sorted arrangement if repeated. However, when asked to verify his hypothesis by writing a set of steps and executing them repeatedly, he quickly realized that it may not lead to the solution. Another pattern students noticed was that each cycle began with the leftmost rod and moved rightward. Students tried to write their algorithm with this abstraction, but without pushing the heaviest stick to the right, their steps wouldn't work.

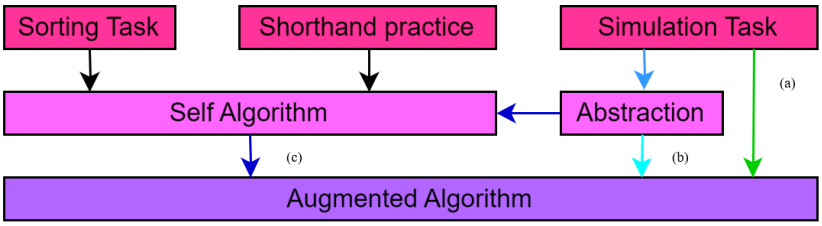

Figure 7. Students' trajectories in the task (a) (green) re-writing the algorithm without much abstraction, (b) (blue, light-blue) writing an algorithm after finding some pattern from the simulation algorithm (partial abstraction), (c) (blue, dark blue) finding a pattern and then applying it to their previous algorithm.

We identified three different pathways via which the students tended to abstract patterns from the simulation algorithm and apply them to their original sorting algorithms (Figure 7). The first involved no abstraction and involved no change to their original algorithm. The second pathway involved partial abstraction and making a few changes to their original algorithm, while the last pathway involved complete abstraction and applying corresponding changes to the original algorithm.

3.2.4 Behavioral Mapping on CT Skills (RQ2)

Based on our overall analysis and the assessment of quantitative and behavioral parameters, some general patterns emerged and helped us classify students as 'good' computational thinkers. This 'goodness' metric evaluates the nature of the students' algorithm (brute force/trial and error vs selection/bubble sort) and their ability to understand that their written solution was not generic. A 'good' computational thinker may think or act slowly, and s/he may require multiple attempts to reach the solution. These patterns, in decreasing order of abstractness, are-

- Persistence: All the participants who did well on the task were not too fazed by their failures. But they weren't disinterested in it either, which was commonly observed in the other students. The apparent failure seemed to push them further. The task had three maximum attempts, but some students wanted to go beyond the given attempts because they wanted to keep trying until they reached a plausible solution. Solution-seeking behavioral patterns (frustration or surprise at failing to find a solution and desire to repeat) and inquisitiveness are common among these students. Hence, we find 'persistence' to map onto the 'iterative' CT skill.

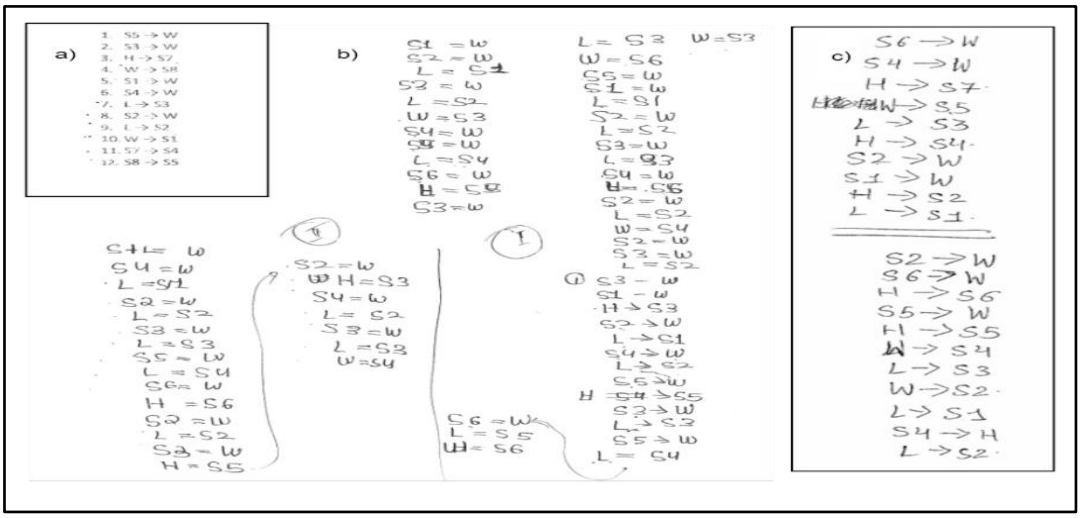

Figure 8. Shorthand notation for writing algorithms: (a) sample provided by experimenter, (b) a 'good computational thinker's attempt to write an algorithm, characterized by persistence and methodological flexibility, and (c) an average student's algorithm.

- Algorithmic thinking: 'Good' computational thinkers eventually settled on a scheme where they would start from the leftmost end, which seemed to show some signs of systematic/algorithmic thinking, unlike other students who seemed to start randomly or on some intuitive understanding of the weights. This could be directly mapped onto the 'algorithmic' CT skill.
- Abstraction: 'Good' computational thinkers were faster at picking up the shorthand notation, i.e. they got the hang of it in fewer steps and did not have to look at the example or think about what to write. Though the shorthand system itself was simplistic, it was hard because it was likely their first formal introduction to such a method of condensed communication. Their ability to pick up the process could thus be a proxy for translation/application ability. This could be directly mapped onto the 'abstraction' CT skill.
- Verification: While verifying the order of the arrangement, students would often check if the first and second rods were correctly ordered and move on to the third and fourth rods (see Figure 5). Very few realized that the order of the second and third rods must also be verified separately. The 'good' computational thinkers performed better than the others on this metric. This mapped onto the 'debugging' CT skill.
- Flexible mindset: 5 out of the 6 'good' computational thinkers were excited about making changes and developing a new algorithm. They all abstracted some patterns from the simulation algorithm provided and tried applying them to their original algorithm to make generic sorting algorithms. This maps onto the 'abstraction' and 'generalization' CT skills.

4. Discussion, Limitations & Implications

This study attempts to uncover the dynamics between academic performance, behavioral dispositions, and performance on CTt. Behavioral tasks were used to identify common practices among 'good' computational thinkers, and assess how their competence reflects in CTt and their academics. We found that students performed significantly better on arrow-

based questions than code block-based ones. Since our students had limited exposure to working with computers, their frequent exposure to ‘arrows’ in their textbooks might have helped them interpret the ‘arrow-based’ questions better than the ‘visual-block’ questions. The higher correlation between Computer Science and English academic performance aligns with earlier findings. Our case studies have revealed subtle differences in different students' methodological and learning approaches and helped us identify different pathways through which they abstracted patterns from given information.

Limited data points constrain our study. Despite the limitations of the study, the method employed has pedagogical value. The task is meant not just as a measurement tool but also as a teaching approach. The lack of correlation between CTt scores and performance in the task does not undermine the validity of the task. The task allows the student to attempt constructing an algorithm, verify it (testing), rephrase it to convey it in simpler language (translation), interact with other algorithms (simulation), and modify their algorithm (debugging): all of these are essential aspects of computational thinking. Educators can use this simple approach to teach children the concepts of algorithms, iteration, testing, debugging, memory allocation, pseudocode, code, etc. without the need for actual computers. Furthermore, sorting has been used extensively as an introductory task for computational thinking pedagogy.

As such, our work adds to the literature that considers CT to be a set of cognitive practices for efficient and effective problem-solving (Shute et. al., 2017). We intend to increase the sample size to gain better insights into CT skills, and incorporate more tasks like searching and navigating into this design.

5. Acknowledgment

We gratefully acknowledge the active participation of all the students and the school authorities who made this study feasible. We also thank the anonymous reviewers who enhanced the scholarship of this work.

References

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International journal of child-computer interaction*, *19*, 30-55.

Caeli, E. N., & Yadav, A. (2020). Unplugged approaches to computational thinking: A historical perspective. *TechTrends*, *64*(1), 29-36.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Detterman, D. K., & Sternberg, R. J. (1993). *Transfer on trial: Intelligence, cognition, and instruction*. Ablex Publishing.

González, M. R. (2015). Computational thinking test: Design guidelines and content validation. In *EDULEARN15 Proceedings* (pp. 2436-2444). IATED.

Guzdial, M. (2008). Education paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25-27.

Kim, B., Kim, T., & Kim, J. (2013). Paper-and-pencil programming strategy toward computational thinking for non-majors: Design your solution. *Journal of Educational Computing Research*, *49*(4), 437-459.

Papert, S., & Harel, I. (1991). Situating constructionism. *constructionism*, *36*(2), 1-11.

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.

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

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, *148*, 103798.

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

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The link magazine*, *6*, 20-23.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st-century problem-solving in K-12 classrooms. *TechTrends*, *60*, 565-568