Computational Thinking Readiness of Incoming High School Students in Taiwan

Greg C. Lee[1 [0009−0002−5624−543X]], Jia-Yi Chen[2 [0009−0003−9929−4263]], and Yu-Wen Yang[3 [0009−0009−4612−3595]]

[1] National Taiwan Normal University, Taipei, Taiwan
leeg@csie.ntnu.edu.tw
[2] National Taiwan Normal University, Taipei, Taiwan
81147002S@ntnu.edu.tw
[3] National Taiwan Normal University, Taipei, Taiwan
wen860806@gmail.com

Abstract. The new national K-12 curricula went into effect four years ago in 2019. Of which, the technology curriculum has shifted towards cultivating Computational Thinking (CT) and programming skills of students. The first group of students who completed middle school education under the new national curriculum entered high school in fall of 2022. A multi-year study is underway to evaluate how the new curriculum has improved students' CT skills. A CT and programming assessment tool was created for this study. Two types of tasks, namely goal-based and problem-based tasks, were designed to test different CT and programming skills. The two goal-based tasks require pattern recognition and generalization CT skills as well as simple repetition and selection programming skills to solve. The two problem-based tasks additionally require students to have good abstraction skills in solving the given tasks. In this paper, results from the first year of on-going study are reported. A total of 17 schools, 130 classes and 4,475 students participated in this study. The incoming high school students were tested during the first four weeks of classes before additional programming lessons were conducted. Thus the result reflects students' learning outcome from middle schools. Overall, the majority of students were able to solve the two goal-based tasks (94% and 92%). However, only less than one fifth of students were able to solve the two problem-based tasks (20% and 10%). This result showed that students need more practice to improve their abstraction skills. Further analysis of students' programs showed that students have the most difficulty in using variables. Findings from this study provide good feedback to middle school teachers. Furthermore, statistical data provides a good baseline for future studies.

Keywords: Computational Thinking · Assessment types and tasks · Online assessment system · CS education

1 Introduction

In recent years, research on Computational Thinking (CT) has focused on the development of effective instructional strategies, the cultivation of foundational

J.-P. Pellet and G. Parriaux (Eds.): ISSEP 2023 Local Proceedings, pp. 167–174, 2023.
https://doi.org/10.5281/zenodo.8431959

skills required, and the formulation of assessment methods and frameworks for CT. In 2019, the new K-12 national curriculum went into effect, which also mandates to have Information technology (IT) classes in each of the two semester of grades 7-9. The new IT curriculum emphasizes on Computational Thinking and programming for problem solving. Thus, the curriculum is more oriented toward computer science than computer applications. Many grades 7-9 teachers in Taiwan, as with many other countries, uses Scratch [7], Blockly [1], MIT App Inventor [4], Greenfoot [3], etc. as computer programming learning tools in practice. Understanding the effectiveness of curriculum implementation is one of the important goals after the implementation of the new curriculum. However, previous assessments of Computational Thinking lack quantitative tools that can be used for a large number of tests, so it is difficult to provide a comprehensive report on the effectiveness of curriculum implementation.

In 2010, Koh, Basawapatna, Bennett, and Repenning [2] developed a visual semantic assessment tool called the Computational Thinking Pattern (CTP) graph particularly for student-created games and simulations. The graph can be used to indicate the existence of CT transferred from games to science simulations. The Fairy Assessment [8] was proposed a few years later in another study to assess two aspects of CT skills, thinking algorithmically, and making effective use of abstraction and modeling. In the assessment, although students were limited to the Alice based learning environment, this study has been regarded as one of the major developments in the assessment of CT skills. In 2015, Moreno-Leon & Robles [5, 6], assessed students' CT skills with "Dr. Scratch" by analyzing students' visual programming projects. Dr. Scratch is an analytical tool that automatically analyzes Scratch projects and assigns a CT score in terms of abstraction, decomposition, parallelism, logical thinking, synchronization, flow control, user interactivity, and data representation. The tool demonstrates how students' programming skills can be improved with the feedback provided.

In summary, there have been some studies evaluating students' CT abilities, some of them assessed students' specific CT abilities by pre-designed tasks, or conversely, directly analyzing students' existing game projects to understand their CT abilities. In this study, we conducted a large scale experiment to assess incoming high school students' Computational Thinking readiness after they have had three years of middle school computer classes. The assessment was based on actual problem solving tasks through programming that require different types of CT and programming skills. In the following sections, the assessment tasks are first explained before the research setup and finding are presented.

2 Chippy Assessment Tool

The Chippy assessment is composed of two types of tasks, with two goal-based tasks to assess students' pattern recognition ability and simple programming ability to perform the same routine repetitively. Students are shown an animation of the task at hand and must recognize the repetitive pattern in the animation before writing Scratch or Blockly code to complete the task. Students can run

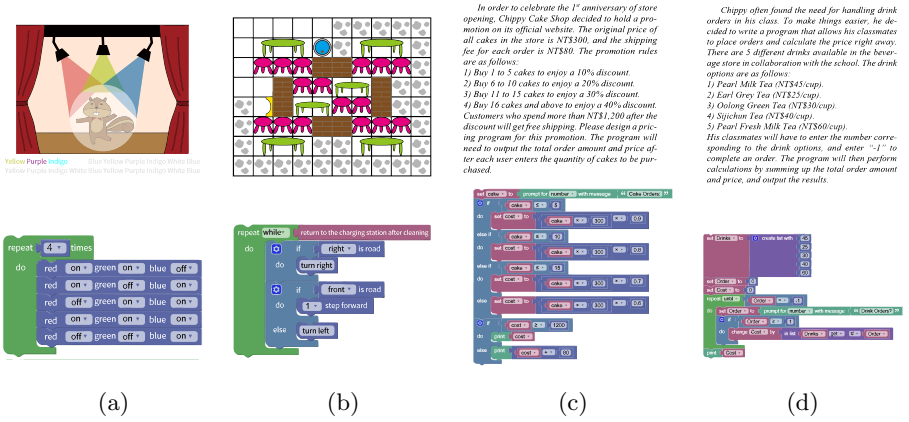
their program and see an animation of the effect of executing their program code step by step. On the other hand, the problem-based tasks are described in words with sample input and output. The tasks are already familiar to the students so are easy for them to understand. The problem-based tasks have test data that correspond to different possible instances/cases of the problem. When students' programs are executed, feedback can then be provided to alert them of different scenarios that have not yet been correctly considered.

The four tasks used in this study are given in Fig. 1. The two goal-based tasks are Light and Robot Vacuum. The Light task repetitively projects different colored light onto the stage. The projected light color is determined by switching on or off the red, blue, and green lights above the stage. Students must watch the animation, observe the repetitive color pattern of the projected lights, and write program to perform the same task. For the Robot Vacuum task, the task animation shows that the robot goes around the classroom to clean dust off the floor. Once students recognize the same turning/moving directions can be used to sweep each quarter section of the classroom, the program code are quite simple as shown in Fig. 1(b).

The two problem-based tasks are Cake Promotion and Drink Orders. The Cake Promotion task asks students to calculate the discount, the shipping fee, and the total price. Students must be able to set proper variables, break problem into taking order and calculation subtasks before designing proper algorithm to solve the problem. The Drink Orders task is similar to the Cake Promotion task but added requirement to use list/array data structure and wrinkle to compute subtotal/total of the order. Both tasks require students to demonstrate good abstraction, problem decomposition and algorithmic design skills. The two tasks have 5 and 6 different possible cases, respectively, that students should consider when designing algorithm. Possible correct programs for the two tasks are shown in Fig. 1(c) and (d). The CT skills needed to complete each task is summarized in Table 1.

Table 1. Computational Thinking skill required for each task.

Skills / Tasks	Pattern Recognition	Abstraction	Decomposition	Algorithm Design
Light	✓			
Robot Vacuum	✓			✓
Cake Promotion		✓	✓	✓
Drink Orders		✓	✓	✓



(a) (b) (c) (d)

Fig. 1. The Chippy Assessment Tasks: (a) Light, (b) Robot Vacuum, (c) Cake Promotion, (d) Drink Orders.

3 Research Setup

3.1 Participants

A call for research participation was made to high school teachers before the start of the fall semester. Teachers with Information Technology classes for incoming high school students (10^{th} grade) were encouraged to participate. In all, 21 teachers from 17 schools with 130 classes for a total of 4,475 students signed up for this study.

3.2 Procedure

The participating teachers use one class period within the first four weeks of the new semester to conduct the Chippy Assessment Test. The assessment was conducted before additional programming lessons in high school. Teachers were asked to use the first 5 minutes of the class to explain the assessment tool. Students then have 45 minutes to work on the above mentioned four tasks. Teachers can help explain the tasks, but students must think, write, and debug programs on their own. Student program for each task was evaluated instantly, and so students know if a task is completed correctly.

3.3 Assessment Scale

The assessment test was scored objectively. As shown in Table 4, Light and Robot Vacuum can only have either 0 or 100 scores, denoting having programmed incorrect or correct problem-solving strategies. For the Cake Promotion and Drink Orders tasks, partial scores were given based on the number of possible distinct scenarios considered and whether they can produce correct output. The maximum score for each task was 100. Students were able to see the evaluation

result instantly. Furthermore, students were given feedback on where or why the program was not given a 100 score. For the goal-based tasks, animation of students' program will reflect the moves as instructed, giving students visual que to where the algorithm failed. For the problem-based tasks, all possible problem scenarios were listed and those that were solved incorrectly were clearly marked.

3.4 Expected Outcome

The maximum score for the assessment test is 400. Having completed the Information Technology curriculum in middle schools ($7^{th} \sim 9^{th}$ grades), high school freshmen (10^{th} grade) are expected to have the competency to complete both goal-based tasks and at least one of the two problem-based tasks with 45 minutes. Students with more practices or can think more quickly can possibly complete all four tasks. Therefore, as shown in Table 2, with each task having a score of 100 points, the **expected** total score of students is between 251 and 350, an equivalent of completing 2.5 to 3.5 tasks. Scores above 351 indicates having **excellent** CT and programming skills. Scores between 151 and 250 indicates not being able to complete one problem-based task and thus only having **moderate** CT and programming skills. Any score below 150 indicates not being able to complete even the two goal-based tasks; therefore, having **inadequate** or **no** CT and programming skills.

Table 2. The CT skills description corresponding to each score range.

Score Range	Relative to the $7^{th} \sim 9^{th}$ grades IT curriculum
351-400	**Excellent** CT and programming skills.
251-350	**Expected** CT and programming skills.
151-250	**Moderate** CT or programming skills.
51-150	**Inadequate** CT and programming skills.
0-50	**No** CT and programming skills.

4 Results and Findings

4.1 Quantitative Analysis

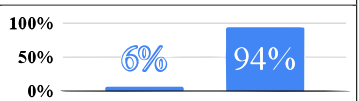
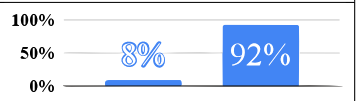
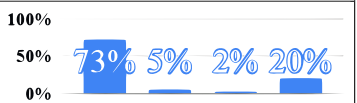
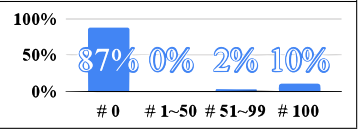
The average score among all 4,475 students participated in this study is 198. Given the expected score of 251 or better, this average score suggests that the CT readiness of incoming high school students, in general, is somewhat below expectation. Table 3 gives the number and the percentage of student scoring in each score range. It can be seen that only 12% (9%+3%) of students performed as expected or better, while most students (71%) exhibited moderate CT and programming skills. It is alarming that close to one fifth (2%+15%) of students still have inadequate CT skills as they enter high schools.

Table 3. Number of students and percentage of students in each score range.

CT/Prog. Skills	No CT	Inadequate	Moderate	Expected	Excellent	Total
Score Range	0-50	51-150	151-250	251-350	351-400	Avg. = 198
No. of Students	74	661	3193	392	155	4475
% of Students	2%	15%	71%	9%	3%	100%

Next, we look at the results by task. Table 4 shows the descriptive statistics. The table shows that majority of students attempted the two goal-based tasks (99% and 96%). Furthermore, majority of students did complete these two tasks (94% and 92%) correctly. This shows that students are capable of finding repetitive patterns from the given problem animation and write programs to perform the same task. For the problem-based tasks, only 56% and 35% of students attempted the two tasks, respectively. This shows that close to half of students either did not attempt or ran out of time to solve problem-based tasks. Of those attempted problem-based tasks, a majority of students (73% and 87%) were not able to receive any partial scores, while only 20% and 10%, for the two tasks, respectively, of student were able to receive full score. These statistical results show that majority of students were not proficient in problem analysis and Computational Thinking, which led to difficulty in solving the problem-based tasks.

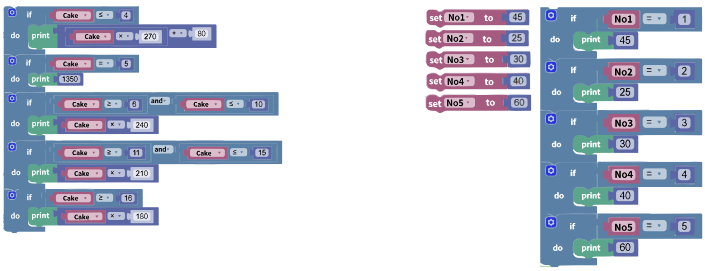
Table 4. Number of people and percentage in each score range.

Task	Attempted	Average Score	Score Distribution
Light	99%	93.6	
Robot Vacuum	96%	92.2	
Cake Promotion	56%	22.7	
Drink Orders	35%	12.0	

4.2 Qualitative Analysis

After looking through students' programs for the problem-based tasks, two observations can be made about student's CT and programming abilities.

1. **Poor understanding and proper usage of variables**
   Although students have learned to use variables, proper usage of variables requires high level of abstraction skill. In general, students do have understanding of storing values in variables, but often can only use variables as constants. For example, in Fig. 2(a), although student's program did use a variable "Cake" to store the number of cakes ordered, that program failed to declare a second variable to keep track of the running total. In this case student did not know how to "update" variable value as required in this task. As another example, in Fig. 2(b), the program had five variables to keep track of the drink prices, but also used the same variables for checking the ordered drink number. Students conceptually associated drink number with drink price in the same variable; thus, unable to use one variable for order checking and another variable for overall cost computation.



(a) (b)

Fig. 2. Sample student program for (a) Cake Promotion, (b) Drink Orders tasks.

2. **Inefficient formulation of different problem instances**
   Another common problem exhibited by students' programs is that the program did not properly condition different cases of the problem with variables. Furthermore, many programs used multiple *if* statements, instead of nested *if-then-else* statement to match natural logical reasoning of different cases. For example, in Fig. 3(a), the program failed to use *if-then-else* structure, but the actual error lay in not being able to keep a running total using a second variable. In Fig. 3(b), in addition to being unable to read in value for variable a and the lack of a variable for running total again, the program used five *if-do* statements to check for drink order number. In both of these examples, students decomposed the problem into a few independent cases. In fact these should be exclusively disjoint cases.

(a) (b)

Fig. 3. Sample student program for (a) Cake Promotion, (b) Drink Orders tasks.

5 Conclusions

In this study, incoming high school students were put to the test to assess their CT and programming learning achievement from their middle school information technology education. The results were rather surprising and alarming. Two recommendations are relayed back to the middle school teachers. First, when training students to think computationally, there is a need to focus more on abstraction of problems, including formulating different problem instances logically. Secondly, more examples and practices are needed to help build conceptualization and good usage of variables.

In summary, this first year study provides a good baseline for future studies. We will continue to conduct this study annually, expanding to more schools and classes. Qualitative results will lead to development of teaching strategies to meet the learning objectives of the information technology curriculum.

References

1. Blockly, https://developers.google.com/blockly. Last accessed 3 June 2023.
2. Koh, K.H., Basawapatna, A., Bennett, V., Repenning, A.: Towards the automatic recognition of computational thinking for adaptive visual language learning. 2010 IEEE Symposium on Visual Languages and Human-Centric Computing. (2010).
3. Kölling, M.: The Greenfoot Programming Environment. ACM Transactions on Computing Education. **10**, 1–21 (2010).
4. MIT APP Inventor, https://appinventor.mit.edu/. Last accessed 3 June 2023.
5. Moreno-León, J., Robles, G.: Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. Presented at the Scratch Conference, 12-15 (2015).
6. Moreno-León, J., Robles, G.: Dr. scratch: a Web Tool to Automatically Evaluate Scratch Projects. Proceedings of the Workshop in Primary and Secondary Computing Education. 132-133 (2015).
7. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: Programming for All. Communications of the ACM. **52**, 60–67 (2009).
8. Werner, L., Denner, J., Campe, S., Kawamoto, D.C.: The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. Proceedings of the 43^{rd} ACM technical symposium on Computer Science Education. (2012).