Evaluating Algorithmic Thinking Ability of Primary Schoolchildren Who Learn Computer Programming

Hidekuni Tsukamoto
Osaka University of Arts
Osaka, Japan
hide1123@osaka-geidai.ac.jp

Yasumasa Oomori
College of Education
Joetsu University of Education
Joetsu, Japan
oomori@juen.ac.jp

Hideo Nagumo
Dept. of Clinical Psychology
Niigata Seiryo University
Niigata, Japan
nagumo@n-seiryo.ac.jp

Yasuhiro Takemura
Dept. of Art Science
Osaka University of Arts
Osaka, Japan
yasuhi-t@osaka-geidai.ac.jp

Akito Monden
Graduate School of Natural Science and Technology
Okayama University
Okayama, Japan
monden@okayama-u.ac.jp

Ken-ichi Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan
matumoto@is.naist.jp

***Abstract*— In this research, a tool for evaluating algorithmic thinking ability of the primary schoolchildren was developed. This tool is based on the three categories of operations used to construct algorithms, namely, sequential operations, conditional branching operations, and iterative operations. Each question in the tool checks to see if the examinee understands the concept of one of the three categories. The tool was developed to evaluate the educational effect of programming education for middle to upper grade (third to sixth grade) primary schoolchildren. Since both Visual Programming Language (VPL) and Textual Programming Language (TPL) could be used, it was required that the tool could be used by both the group of children who use VPLs and the group of children who use TPLs. To make it possible, no programming language appeared in the questions in the tool. The teaching materials for the programming education were also developed in such a way that the three basic concepts of algorithm, namely, sequential processing, conditional branching, and repetitive processing, were clearly taught. The target VPL in this research was Scratch. The evaluation tool was conducted in a weekend class of programming education for primary schoolchildren, and the algorithmic thinking ability of the schoolchildren was analyzed.**

Keywords—programming education; primary education; algorithmic thinking; evaluation tool

I. INTRODUCTION

Recently, it has been recognized that introducing computer programming to young people is imperative [1, 2], and as a result, programming has been introduced to the primary school curriculum in many countries, including Japan [3-7]. According to the Japanese New Course of Study which will be introduced in 2020, Japanese primary schoolchildren will be required to learn computer programming as an interdisciplinary element appearing throughout the curriculum [8]. In preparation for this change, many after-school or weekend classes of programming education for primary schoolchildren are conducted by educational organizations [9].

In many of those programming classes, the participants are typically divided into teams of 4 to 6 children, and are supposed to program attractive computer games with the help of teaching assistants. In most cases, the teams present the games they programmed at the end of the course. In the webpages of the programming classes or in the newspaper articles about the classes, positive comments of the children who participated in the classes can be found. They express feelings of enjoyment and senses of satisfaction. Capturing children's attention is definitely important when teaching programming to them. However, that is not enough, and fostering learning that suits the goals of the programming education is also important [1].

The goals of programming education for K-12 nowadays are often discussed in terms of Computational Thinking (CT) [10,11]. CT is the term introduced by Wing in the article [12]. She stated that CT included a range of mental tools that reflected the breadth of the fields of computer science, and that it involved solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. There are high levels of interest in developing CT skills among schoolchildren in many countries. However, there are several issues that have to be addressed for the effective integration of CT in K-12. One of them is how to define CT as a key 21st century skill for schoolchildren [11]. Since the statement of CT in the article [12] is very abstract, and not easy to be implemented in practical school activities, several researchers have expressed more concrete definitions of CT. Among them, Barr et al. [10] defined nine core CT concepts: Data collection, Data analysis, Data representation, Problem decomposition, Abstraction, Algorithms and procedures, Automation, Parallelization, and

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

Simulation. These concepts are concrete and easier to implement and assess.

Though CT has become very popular among the researchers working on programming education for K-12, only a limited amount of research has been carried out on assessing children's grasp of CT concepts [11]. It is partly because there has not been concrete definitions of CT. Without clear definitions of CT, it is not possible to make assessment tools that check to see if the examinees have gained CT ability. Another reason is that it is difficult to hold a programming class for primary schoolchildren with a sufficient number of children, and to collect data from the children. It is difficult especially in Japan because those programming classes may waste the time of accompanying guardians, and also because collecting data from the children requires the guardians consent.

The purpose of this research is to evaluate educational effects of programming education for primary schoolchildren. For this purpose, we hypothesize that programming education fosters CT. This hypothesis is supported by the statement "programming can make CT concepts concrete and become a tool for learning" in [11]. From this hypothesis, we can state that our goal of programming education for primary school children is to foster CT. And once the goal is determined, it is possible to devise how to assess educational effects of programming education. Since CT definitions are still somewhat forthcoming [13], the nine core CT concepts defined by Barr et al. [10] were adopted in this research. Among the nine concepts, only one concept, namely, "Algorithms & procedures" was chosen for the assessment in this research because the time for the schoolchildren to take a test for evaluating their skills was limited. Therefore, in this research, we propose assessment tools that can be used to assess algorithmic thinking ability of the schoolchildren.

II. Related Works

Morrison et al. [14] reported on the effects of subgoal labels on the programming education for the students in technical universities. They used Parsons problems, in which correct code is broken into code fragments that have to be put in the correct order with the correct indention, to measure the problem solving performance of their students. The students in the class were given worked examples of the programing codes in the three formats: with no subgoal labels, with meaningful subgoal labels, and with the spaces for writing subgoal labels the students were supposed to fill. The students were also divided into a contextual transfer group and an isomorphic problem group. They found that the students who were given subgoal labels performed better than those who had no subgoal labels or who generated their own subgoal labels. Though they assessed the effects of subgoal labels on the programming education, the target groups were those of university students, not primary school children.

Weintrop et al. [15] compared conceptual understanding in blocks-based programming languages and text-based introductory programming languages. For this purpose, they developed assessment tools which could be used for both blocks-based languages and text-based languages. The tool consisted of multiple-choice questions which asked the results of the program printed on the sheet. The questions were designed commutative in that the programs shown in the sheets could be both in block-based and text-based. After collecting data from the programming education to the 90 high school students, they found more misconceptions on text-based versions of questions than block-based versions. Though they developed assessment tools and used them in the experiments, the target students were those in high schools, not in primary schools.

Rodrigues et al. [16] investigated the relation between the CT skills of high school students developed by computer programming and the academic performance and problem solving skills of the students. To assess the CT skills, they measured the students' knowledge of computer programming. To assess the academic performance of the students, they used ENEM (National Exam of High School), and to assess the problem-solving skills they used WASI (Whimbey Analytical Skills Inventory). From the analysis, in the context of basic education, they found that students trained in CT topics perform better in problem solving, Mathematics and Natural Sciences as compared to students who did not develop CT skills. Though they measured the CT skills of the students, they did not study the effect of programming education. Rather, they studied the relation between programming skill and problem solving skill, and that between programming skill and academic performance. And again, the target students were those in high schools, not in primary schools.

Oliveira et al. [17] studied the relation between the student's ability to compute and his/her academic performance in a primary school. They developed a model of computation based on the Turing Machine model, which had been named DMM (Drawing Machine model). The machine of the DMM operated on a grid paper by interpreting a drawing-based language and producing a result. A total of 81 students attending a primary school participated in the survey. The authors employed statistical tools to calculate the correlation between the test results and the students' academic performance in school. There were strong and moderate correlations between the ability to compute, as measured by the proposed experiment, and student performance in the school surveyed. Though this paper described the survey on primary schoolchildren, it did not discuss the effect of programming education for primary schoolchildren.

Williams et al. [1] described the design and implementation of a computer micro-world game designed to introduce the core constructs and techniques of computer programming. They conducted several interactive sessions with a group of 11 primary schoolchildren using the game they designed. They conducted pre and post surveys asking questions such as "I want to learn more about programming computers," with the five point Likert scale. The results showed that the game sessions had a positive impact on the students' confidence in their knowledge of computers, as well as, their thinking of programming as a fun experience. Though the authors taught programming to primary schoolchildren and collected data from the students before and after the session, the data collected were feedback from the children, and not the data about the skills and the level of understanding of the children.

When computer programming is taught to young children, it is often taught as a part of Computer Science (CS) education, and often accompanied with CS Unplugged teaching materials. The work on CS Unplugged started in the 1990s by Bell et al. [18, 19], and was used to develop experience in communicating computer science without using computers. Some educational groups around the world provide collections of free learning activities for CS Unplugged in the form of games and puzzles and others [20].

III. Assessment Tool

In this research, a tool for evaluating algorithmic thinking ability of the primary schoolchildren was developed. This tool is based on the three categories of operations used to construct algorithms, namely, sequential operations, conditional branching operations, and iterative operations. Each question in the tool checks to see if the examinee understands the concept of one of the three categories.

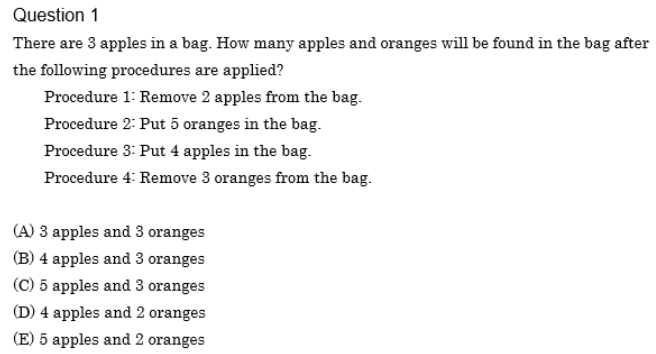
A. Requirement for the Tool

The tool was developed to evaluate the educational effect of programming education for middle to upper grade (third to sixth grade) primary schoolchildren. Since both Visual Programming Language (VPL) and Textual Programming Language (TPL) may be used, it was required that the tool could be used by both the group of children who use VPLs and the group of children who use TPLs. To make it possible, no programming language appeared in the questions in the tool. The teaching materials for the programming education were also developed in such a way that the three basic concepts of algorithm, namely, sequential processing, conditional branching processing, and repetitive processing, were clearly taught. The target VPL in this research was Scratch.

The assessment tool was supposed to be conducted twice, namely, before and after the programming class. The time duration of the programming classes vary depending on the organizing institutions, and is typically between two hours and five hours. Therefore, the time we can spend for pre-test and post-test is ten to fifteen minutes each. Within this time interval, the children are supposed to solve three questions, corresponding to sequential operations, conditional branching operations, and iterative operations.

B. The First Draft of the Tool

We produced the first draft of the tool based on the requirements mentioned above. We chose multiple choice questions for the first draft because it would be easier to grade the answers. The first problem checks to see if the examinee understands the concept of sequential operations. Fig. 1 shows the first draft of the first problem. Though the question in the figure is written in English, the actual question was written in Japanese, and this is true for other figures. In this problem, at first, the initial condition of a bag, namely, how many apples and oranges are in the bag, is given. After that operations that change the number of apples and oranges in the bag are listed up sequentially. The examinees are supposed to answer the final number of apples and oranges in the bag.


Question 1

There are 3 apples in a bag. How many apples and oranges will be found in the bag after the following procedures are applied?

Procedure 1: Remove 2 apples from the bag.
Procedure 2: Put 5 oranges in the bag.
Procedure 3: Put 4 apples in the bag.
Procedure 4: Remove 3 oranges from the bag.

(A) 3 apples and 3 oranges
(B) 4 apples and 3 oranges
(C) 5 apples and 3 oranges
(D) 4 apples and 2 oranges
(E) 5 apples and 2 oranges

Fig. 1. First draft of question 1

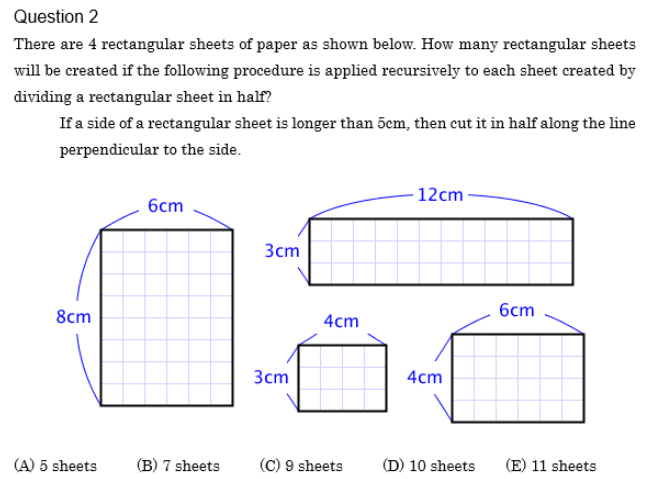


Fig. 2. First draft of question 2

The second problem checks to see if the examinees understand the concept of conditional branch operations. Fig. 2 shows the first draft of the second problem. Four rectangle sheets of paper appear in the problem. The examinees are supposed to check if a side of any rectangle sheet is longer than a certain value. If that is true, then the examinees are supposed to cut the rectangle sheet in half along the line perpendicular to the side. (Or they could draw a line along which the rectangle is cut.) The examinees are supposed to recursively apply this procedure to the smaller rectangle sheets created by cutting a rectangle sheet.

The third problem checks to see if the examinees understand concept of iterative operations. Fig. 3 shows the first draft of the third problem. In this problem, the examinees are supposed to select a pattern of X marks generated by iteratively perform blocks of operations. Here, a block of operations is expressed as sentences of operations inside a rectangle.

When we conducted pilot experiment of the first draft of the tool to three primary schoolchildren (two sixth grade boys and one fifth grade boy), we found the following problematic points.

- The time interval allocated for the pre-test and post-test (15 minutes) was not enough for the children to solve all the problems in the test.

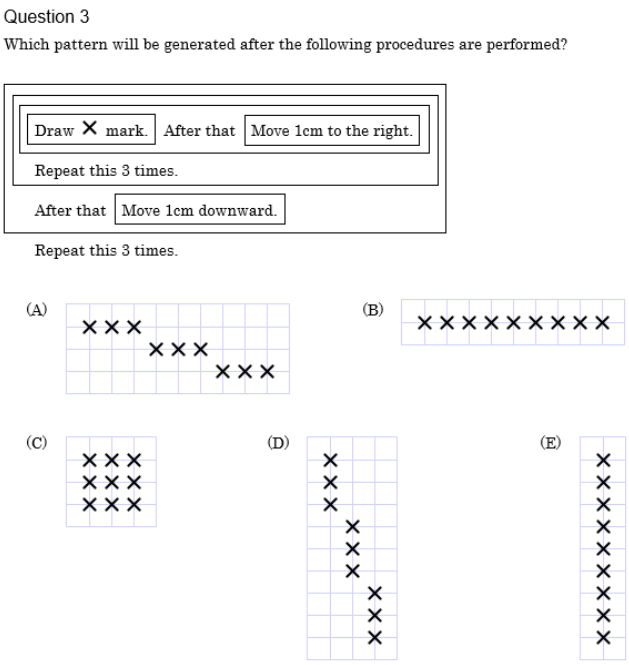


Fig. 3. First draft of question 3

- It was difficult for the children to understand the meaning of the instruction of the problems, especially, that of the problem of conditional operation.
- Since the post-test was exactly the same as pre-test, some children remembered the answers of pre-test and just wrote them down without trying to solve the problems.
- Since the answers were multiple choice, the process of answering could not be seen.

Actually, the shortage of time interval for the test was related to the difficulty of understanding the instructions of the problems. In the first draft of the problems, we asked the children to recursively cut a rectangle in half to produce two congruent rectangles if a side of a rectangle was longer than a particular value. Since we tried to eliminate any ambiguity in the instructions, the sentences became long and complicated. It would be okay if the problems were given to middle school students, but not the children in primary schools. Since the complexity of the wording was inherent to the problem itself, we decided not to use the same problem, and replaced it with a different problem of which the instruction could be composed of simpler sentences.

In order to remediate the problem of the children remembering their answers in the pre-test and writing them down in the post-test, some details of the questions were changed so that the answers of the pre-test and post-test were not exactly the same, but the difficulty levels of them were the same.

C. *The Revised Tool*

Fig. 4 shows the revised first problem which checks to see if the examinees understand the concept of sequential

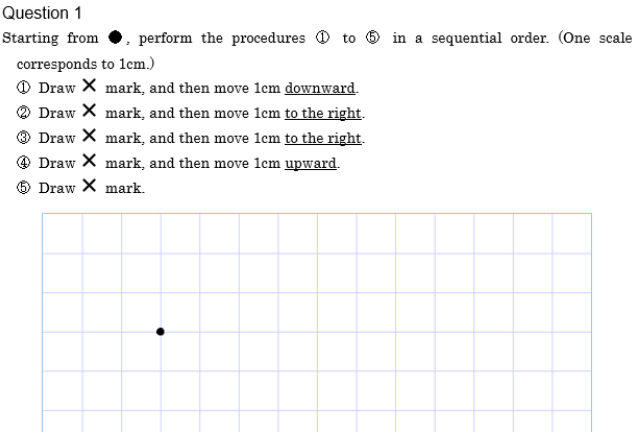


Fig. 4. Question to check if the examinee understand the concept of sequential processing

operations. In this problem, five numbered procedures are given, and the examinees are supposed to perform these procedures in a sequential order. If an examinee answers correctly, he/she would draw a certain pattern of X marks on the grid.

Fig. 5 shows the revised second problem which checks to see if the examinees understand the concept of conditional branching operations. There are many colored walls drawn as red, green, and blue thick bars in the working area of the problem. The red walls are also labeled with "r", green walls with "g", and blue walls with "b". In the actual worksheets, Kanji characters that represent colors are used as the labels. The examinee is supposed to start drawing a line from the black dot in the direction of an arrow, and extend the line until

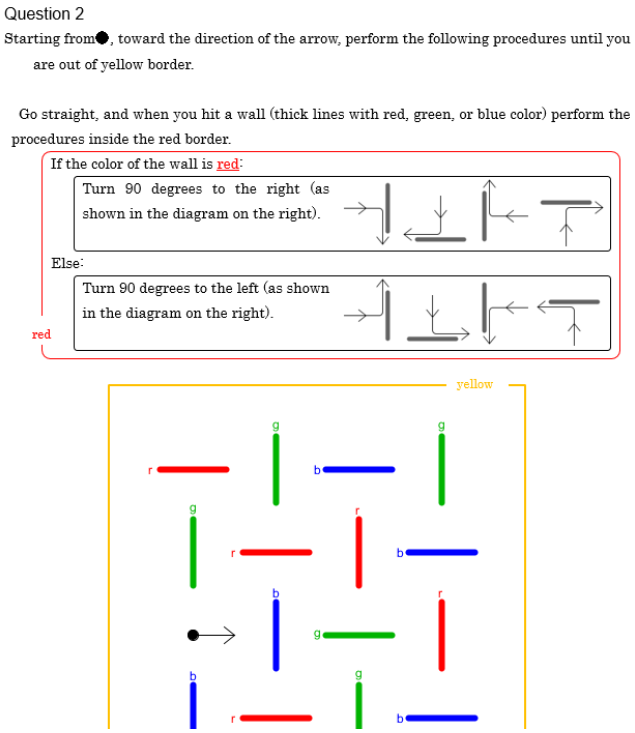


Fig. 5. Question to check if the examinee understand the concept of conditional branching

it meets a wall or goes out of a border line. When the line meets a wall, the examinee has to check the color of the wall. If the wall is red then he/she has to change the direction of the line 90 degrees clockwise and keep on extending the line. Otherwise, he/she has to change the direction of the line anticlockwise and keep on extending the line. He/she should stop extending the line if it goes out of the border. In this problem, the instructions were made as simple as possible, and as short as possible so that the children did not take time to interpret the instruction. Since the terms "right" and "left" could make children get confused sometimes, we provided diagrams to make sure children understand the meaning of "Turn 90 degrees to the right" and "Turn 90 degrees to the left" in Fig. 5. Also, a colored rectangular border was used to clearly indicate which rectangular border was mentioned in the explanation.

Fig. 6 shows the revised third problem which checks to see if the examinees understand the concept of iterative operations. Here again, rectangular borders, sometimes with colors, were used to surround a block of procedures so that a cluster of procedures was clearly visible and the nesting structure of blocks of procedures was also clearly visible. When expressing iterative operations in Japanese, a problem arose. That is, in Japanese, verbs are usually placed at the end of sentences, while they are placed just after the subject of sentences in English. Because of this reason, the sentence to express the number of iteration was placed after the block to be iterated, in the description of the algorithm.

D. Grading Procedure

Each question was graded out of five points so that the maximum score of the test was 15 points. When grading any of the three questions, one point was given to the score for each correct operation with a ceiling of five, and after that, one point was deducted from the score for each wrong operation, but with a ceiling of two. If the result was less than zero, zero was given as the final score. We did not deduct points for a misunderstanding of lengths so long as the misunderstanding was consistent. For example, we did not deduct points if he/she consistently moved forward 2cm when he/she was supposed to move forward 1cm. Also, if a loop is encountered in Question 2, then only one cycle is used in the grading process.

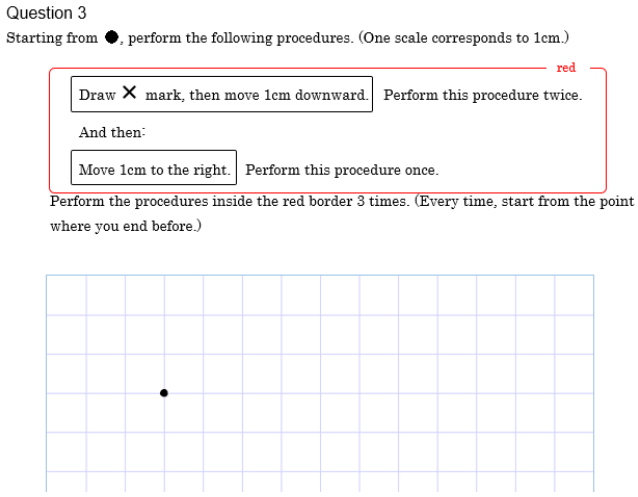


Fig. 6. Question to check if the examinee understand the concept of repetitive processing

Some examples of answers to the questions in the assessment tool are shown in Fig. 7. In Fig. 7, the image on the right side of "Q1" is an example of answers to Question 1 shown in Fig. 4. In this example, the first five X marks form a correct pattern, but many extra X marks are added after that. When grading this answer, we gave five points to the correct five X marks first, and then deducted two points to make the score to the answer three. The reason for deducting only two points when there were many more extra X marks is that the ceiling for the points for the wrong operations was two.

The image on the right side of "Q2" is an example of answers to Question 2 shown in Fig. 5. For this example, we added five points to the score, and then deducted two points to make the score three. The first five points added to the score came from the number of correct turnings at the walls (first, second, third, fourth, fifth, seventh). Though there were six correct turnings, since the ceiling for the number of correct operation is five, we gave five to the score. The second two points deducted from the score came from the number of wrong turnings at the walls (sixth, eighth, ninth, tenth). Though there were four wrong turnings, since the ceiling for the number of wrong operations is two, we deducted three points from the score.

The image on the right side of "Q3" is an example of answers to Question 3 shown in Fig. 6. In the leftmost column, the top two X marks were placed with correct operations, but the bottom one was not. This is also true for second and third column. Therefore, we added five points to the score and deducted two points from the score because, at this point of

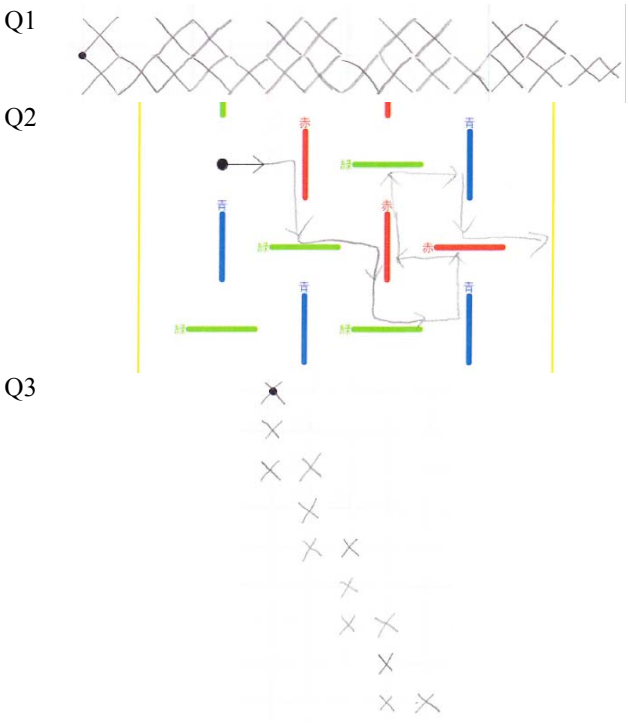


Fig. 7. Examples of the answers to the questions.

time, there are more than five X marks placed with correct operations and more than two X marks placed with wrong operations. As a result, the score for the answer is three.

IV. Experiment

An experimental weekend class of programming education for primary schoolchildren was conducted using a VPL. Five primary schoolchildren, one third grade girl, two fourth grade girls, and two sixth grade boys participated in the class. Unfortunately, there was not a chance to conduct weekend classes using TPLs. The curriculum of the weekend class is shown in TABLE I. As shown in the table, the three basic categories of algorithm were taught with unplugged teaching materials and also with Scratch before game programming was introduced to the schoolchildren. The unplugged teaching materials used in the first teaching session to teach sequential and iterative operations were originated from Code.org webpage [20]. We used "Graph Paper Programming" for teaching sequential algorithm, and "Getting Loopy" for teaching iterative algorithm.

Fig. 8 shows the programming assignment for learning sequential operation. In this assignment, the learners are supposed to write programs so that the sprite of Cat starts from the sprite of letter S and moves through all the sprites of letter R (usually there are multiples of them) and finishes at the sprite of letter G in the stage located on the upper left side of the window. When working on the assignment, at first, the learners position the sprites of letters by selecting a sprite of letter in the Sprite List area on the lower left side of the window, and click on the green flag in the Scripts area. By doing so, the sprite selected is located randomly on the stage. In the case of letter S and letter G, only one sprite for each letter is placed on the stage, but more than one sprite of letter R can be located on the stage. The learners are supposed to write programs for the sprite of Cat, and can use only three command blocks: moving forward the sprite of Cat one unit length and wait, turning 90 degrees clockwise, and turning 90 degrees anticlockwise.

TABLE I. Curriculum of the Weekend Class

Time	Contents
12:50-13:05	Pre-test
13:05-14:00	Sequential and iterative operations with unplugged teaching materials
14:10-15-15	Sequential, iterative, and conditional branching operations with Scratch
16:15-16:05	Game programming with Scratch
16:05-16:20	Post-test

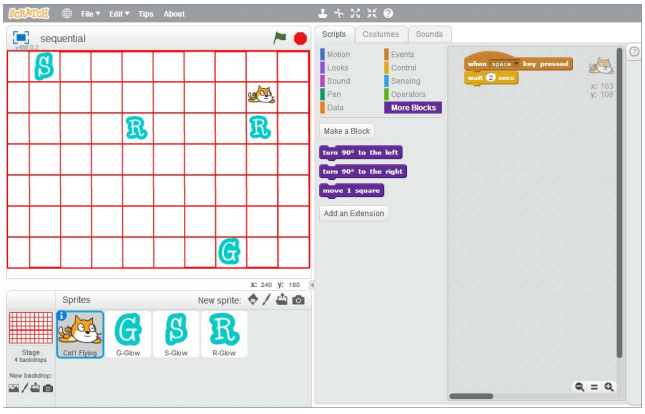


Fig. 8. Assignment for learning sequential operation

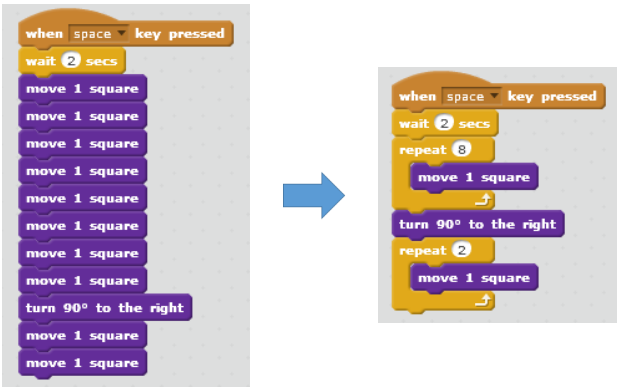


Fig. 9. Assignment for learning iterative operation

The same setting is used to learn iterative operations. When programming the movement of the sprite of Cat, multiple numbers of "move forward and wait" block can be stacked as shown in Fig. 9. In such cases, the learners are encouraged to use REPEAT block to make the script shorter.

Fig. 10 shows the programming assignment for learning conditional branching operation. There are three rows where a sprite of Cat and that of Mouse is placed. At first, the sprites of Cat are visible but those of Mouse are not. When the green flag in the Scripts area is clicked, three sprites of Mouse are placed in the stage in such a way that one sprite of Mouse is on the right of one sprite of Cat, but the distance of the two are randomly determined. The assignment to the learners is to write programs in such a way that each sprite of Cat moves to the right and when it coincides with a sprite of Mouse, it says "I've found a mouse". The trick here is that the position of the third sprite of Mouse change its position when the learner starts running his/her program so that the program does not work properly without using conditional branching block.

After learning these three basic concepts of algorithm, learners were supposed to write a program of a simple shooting game. The algorithmic thinking ability was assessed using the tool introduced in the previous section before and after the

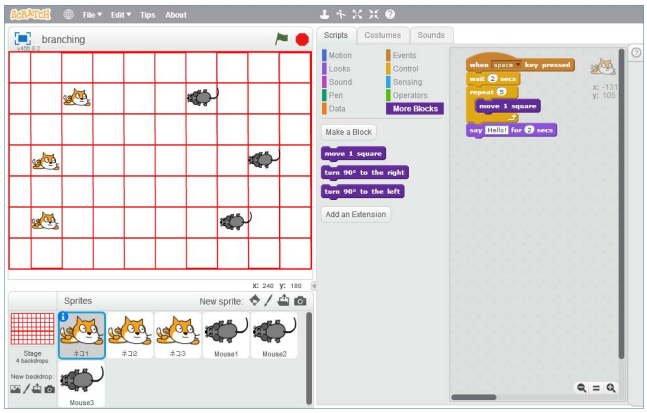


Fig. 10. Assignment for learning conditional branching operation

programming sections.

V. Analysis

The results of pre-test and post-test are shown in TABLE II and Fig. 11. In TABLE II, means and variances of the questions in the pre-test and post-test, as well as the total scores of the two tests are shown. In Fig. 11, the corresponding results of pre-test and post-test are compared with boxplots. In each plot, the box on the left is for pre-test and that on the right is for post-test.

By observing Fig. 11 and TABLE II, it appears that mean for each question and total score increased from pre-test to post-test. However, since the number of participants is only five, there is not much statistical meaning for this. (It is not possible to accept or reject the hypothesis based on this data.) When paired t-test was conducted, statistically significant differences were not observed for any pair of corresponding questions or total scores. Another observation is that the variances of post-test is always smaller than those of pre-test. This result may indicate that there were children who could not get a high score in the pre-test, but could get a high score in the post-test.

When giving an assignment of algorithm to primary schoolchildren and making them understand the steps to follow, there are always problems of language. Writing an algorithm in such a way that every schoolchild understands it is not a trivial matter. Especially, it is difficult for the schoolchildren to understand the concept of *block* in which multiple instructions are put, because blocks can be repeated, executed as a whole if a condition is met, or nested in another block which again can be repeated. It could be that the children found the questions in the pre-test difficult because of this reason. This hypothesis cannot be confirmed here, because no control group was provided in this research.

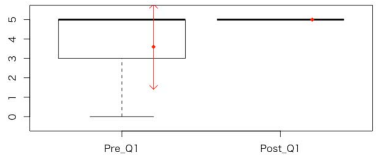
VI. Conclusions

In this research, an assessment tool for assessing algorithmic thinking ability of primary schoolchildren was proposed. There were three questions in the tool, and they check if the examinees understood the basic concepts of algorithms, namely, sequential operations, conditional branching operations, and iterative operations. The tool was used for pre-test and post-test. The questions for post-test were slightly changed from those for pre-test so that the two tests were different but the level of difficulty were the same.

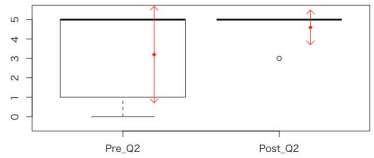
This assessment tool was used in a weekend programming class for primary schoolchildren. There were five participants, one third grade girl, two fourth grade girls, and two sixth grade boys. It appeared that mean for each question and total score increased from pre-test to post-test. Also, it appeared that the variances of post-test were always smaller than those of pre-test. However, since the number of participants is only five, there is not much statistical meaning for this.

TABLE II. Curriculum of the Weekend Class

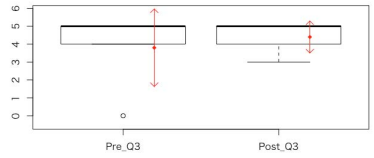
	Pre				Post			
	Q1	***Q2***	***Q3***	***Total***	***Q1***	***Q2***	***Q3***	***Total***
mean	3.6	3.2	3.8	10.6	5.0	4.6	4.4	14.0
variance	4.8	6.2	4.7	39.3	0.0	0.8	0.8	1.0



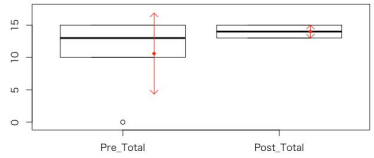
Question 1



Question 2



Question 3



Total score

Fig. 11. Results of pre-test and post-test

Due to the difficulty of obtaining data from primary school children, enough data could not be collected. Also, data from control group could not be collected. As future work, we plan to conduct more weekend classes of programming education for primary schoolchildren using both VPLs and TPLs so that we can collect a sufficient amount of data.

Although it is crucial to teach schoolchildren computation and incorporate programming, we have to recognize that there is some psychological research explaining that exposing children to technological devices might become a problem to the children who can become addicted to the use of these devices [21].

Acknowledgment

This work was partly supported by Grants-in-Aid for Scientific Research (C) No.JP16K01087 from Japan Society for the Promotion of Science.

References

[1] C. Williams, E. Alafghani, A. Daley Jr., K. Gregory, and M. Rydzewski, "Teaching programming concepts to elementary students," in Proc. IEEE FIE 2015, 2015, pp. 706-714.

[2] C. Duncan, T. Bell, and S. Tanimoto, “Should your 8-year-old learn coding?,” in Proc. ACM WiPSCE’14, 2014, pp. 60-69.

[3] F. Heintz, L. Mannila, and T. Färnqvist, “A review of models for introducing computational thinking, computer sciece and computing in K-12 education,” in Proc. IEEE FIE 2016, 2016.

[4] White House, “Computer science for all,” 2016. https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all.

[5] GOV.UK, “National curriculum in England: computing programmes of study,” 2013. https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study.

[6] Australian CURRICULUM, “Digital Technologies,” 2015. http://www.australiancurriculum.edu.au/technologies/digital-technologies/curriculum/f-10?layout=1.

[7] Ministry of Education, Culture, Sports, Science and Technology Japan, “Programming education in primary school level,” (in Japanese) 2016. http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm.

[8] Ministry of Education, Culture, Sports, Science and Technology Japan, “Summary report of ongoing deliberations on the new course of study in Japan,” (in Japanese) 2016. http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/004/gaiyou/1377051.htm.

[9] Public Management Ministry Japan, “Research on the basic act on human resource development of computer programming (report),” (in Japanese) 2015. http://www.soumu.go.jp/main_content/000361429.pdf.

[10] V. Barr and C. Stephenson, “Bringing Computational Thinking to k-12: What is involved and what is the role of the computer science education community?,” ACM Inroads, vol. 2, no. 1, pp. 48-54, 2011.

[11] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, and K. Engelhardt, “Developing Computational Thinking in compulsory education,” European Commission, JRC Science for Policy Report, 2016. http://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf.

[12] M. Wing, “Computational Thinking,” Viewpoint, Communications of the ACM, vol. 49, no. 3, pp. 33-35, 2006.

[13] A. Repenning, “Programming goes back to school,” Viewpoint, Communications of the ACM, vol. 55, no. 5, pp. 38-40, 2012.

[14] B. Morrison, L. Margulieux, B. Ericson, and M. Guzdial, “Subgoals help students solve Parsons problems,” in Proc. ACM SIGCSE’16, 2016, pp. 42-47.

[15] D. Weintrop and U. Wilensky, “Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs,” in Proc. ACM ICER’15, 2015, pp. 101-110.

[16] R. Rodrigues, W. Andrade, and L. Sampaio Campos, “Can Computational Thinking help me? A quantitative study of its effects on education,” in Proc. IEEE FIE 2016, 2016.

[17] O. Oliveira M. Nicoletti, L. Cura, “Quantitative correlation between ability to compute and student performance in a primary school,” in Proc. ACM SIGCSE’14, 2014, pp. 505-510.

[18] M. Fellows, T. Bell, and I. Witten, “Computer Science Unplugged. Computer Science Unplugged,” 2002. available at http://www.lulu.com.

[19] T. Bell, P. Andreae, and A. Robins, “A case study of the introduction of computer science in NZ schools,” Trans. Comput. Educ., vol. 14, no. 2, pp. 10:1-10:31, 2014.

[20] Code.org, “Anybody can learn | Code.org,” https://code.org/

[21] K. G. Burris and C. Wright, “Review of Research: Children and Technology: Issues, Challenges, and Opportunities,” Childhood Education, vol. 78, no. 1, pp. 37-41, 2001.