Informatics in Education, 2024, Vol. 23, No. 1, 145–178

DOI: 10.15388/infedu.2024.07

Effect of an OwlSpace Programming Course on the Computational Thinking of Elementary School Students

Wei-Ying LI, Tzu-Chuen LU*
Department of Information Management, Chaoyang University of Technology
Taichung 41349, Taiwan
e-mail: totoro@y-shun.com.tw, tclu@cyut.edu.tw

Received: January 2023

Abstract. This study investigates the effect of programming courses on the computational thinking (CT) skills of elementary school students and the learning effectiveness of students from different backgrounds who are studying programming. We designed a OwlSpace programming course into an elementary school curriculum. Students in fourth and fifth grades were taught the fundamentals of programming. We measured and analyzed the effectiveness of their CT skills and self-efficacy in CT. The researchers analyzed the changes in the CT of different gender, different grade, and different past experience students in programming courses and then made specific recommendations for information technigy teachers and related units. The results demonstrate that students learned and improved their CT skills by taking OwlSpace programming course. Additionally, gender, grade, and past experience are found to have no impact on the students' learning that means the course can improve students ability without limited any characteristics.

Keywords: computational thinking, OwlSpace programming, script-block-based programming language.

1. Introduction

With the advent of the global information age, information technology products, ranging from simple web pages to mobile applications, the Internet of Things (IoT), and big data to artificial intelligence, have become the most common and convenient tools available to us. These useful tools comprise coded instructions and rely on the collaboration of multiple programs to power various innovative electronic products. To develop relevant information products and enable information equipment to operate

* Corresponding author

as intended, the work of programmers is particularly critical. Various countries around the world have developed plans and made considerable efforts to cultivate talents with programming-related skills and knowledge. For example, to address the shortage of information technology professionals, many countries have added computer science-related courses to K-12 education, so that children can be trained in computational thinking (CT) at a young age (Lindberg *et al.*, 2018).

In 2006, Wing (Wing, 2006) stated that CT is a method of solving problems by applying core concepts of computer science. CT is not limited to computer science; rather, it is a necessary skill that everyone should possess. Wing also believed that learning CT in early childhood can have a positive effect on studying other subjects (Wing, 2008). Most elementary school students have easy access to various electronic and mechanical products. In addition, students can use computing devices at home and school on a regular basis, allowing them to use computing tools with ease. The sooner schools provide relevant education, the sooner CT can begin to develop. (Wing, 2008).

Programming education is one of the most effective ways to develop CT (Resnick *et al.*, 2009). The most effective way to develop CT during early childhood is to incorporate programming into school curricula (Moreno-León, 2018). Cynthia and Woollard (Cynthia and Woollard, 2014) believed that there is a correlation between programming activities and CT capabilities, and that a complete CT element can be cultivated during the programming learning process, including abstraction, decomposition, algorithmic design, evaluation, and generalization. However, because script-based programming primarily involves abstract concepts, this learning method is challenging for many students and can cause fear and disinterest in learning (Resnick *et al.*, 2009). Therefore, block programming tools, such as App Inventor (Wolber *et al.*, 2015), Alice (Cooper *et al.*, 2000), and Scratch (Resnick *et al.*, 2009), have been used in the past to facilitate the development of CT in young learners through highly accessible programming activities. To make abstract concepts easier to understand while teaching programming skills, block programming software intuitively demonstrates the operation process of a program using visualization and animation techniques (Wing, 2008).

Although block-based programming languages can more intuitively introduce programming concepts to beginners, script-based programming languages are better suited to challenging learners because they allow for constructing comparatively more complex programs. When researchers investigated the impact of block-based programming languages on students, students stated, "Block-based programming languages seem simple and have little to do with real programming." Students do not seem to associate the formal coding process with tasks performed using block programming tools (Parsons and Haden, 2007). Students also identified some disadvantages of block-based programming languages, including a lack of realism and less powerful features (Weintrop and Wilensky, 2015). In addition, interactive tools such as Scratch are inadequate to help students with the transitions. Therefore, transition from block-based programming languages to more advanced programming languages may be more challenging than expected (Chetty and Barlow-Jones, 2012).

This study designed a script-block-based programming language called OwlSpace programming language, which combines the beneficial elements of both script-based

and block-based programming languages, to simplify the process of transitioning students from block-based programming languages to script-based programming languages. It also eases the transition of students from block-based to script-based programming languages. The interface and operation of the proposed OwlSpace programming language are similar to that of a block programming language, which may prevent students from developing a fear of abstraction. Furthermore, OwlSpace is an open-source language that combines the interactivity and syntax of "scripting" languages, such as Python, Matlab, and R, but with the speed of "compiled" languages such as Fortran and C. In scripting languages such as Python, the user types the code line by line into the editor, and the language interprets and runs it, then returns the result immediately; however, in languages such as C and Fortran, code must be compiled before it can be executed. Scripting languages such as Python are easier to use, whereas languages such as C and Fortran produce faster code. As a result, programmers often develop algorithms in a scripting language, then translate them into C or Fortran. OwlSpace can solve the two language problems because it runs similar to C but reads similar to Python.

Additional information about OwlSpace is available on the website: `https://www.sdc.org.tw/product/貓頭鷹-steam-程式設計平臺owlspace-貓頭鷹創作空間-貓頭鷹線/`.

The system can be download from `https://tinyurl.com/owledu2208`.

2. Related Works

2.1. *Cognitive Load*

When people are performing a task, their attention and cognitive resources are limited, and the use of these resources is referred to as "cognitive load. Sweller defined cognitive load as the amount of load imposed on an individual's cognitive system when performing a specific task (Sweller, 1988). If a task is too complex or requires cognitive resources beyond an individual's capacity, then a higher load will be imposed. This can lead to decreased task performance, errors, or increased fatigue. For learners, cognitive load refers to the load imposed on the learning system when a specific task is added (Paas and Van Merriënboer, 1994). More specifically, cognitive load theory assumes that any learning task imposes three types of cognitive load on working memory: intrinsic cognitive load, extraneous cognitive load, and germane cognitive load (van Merri€enboer and Sweller, 2005; Paas *et al.*, 2003; Paas *et al.*, 2004; Sweller *et al.*, 2011). Intrinsic cognitive load – This is the inherent cognitive load of the instructional material, determined by its complexity and difficulty. Extraneous cognitive load – This refers to cognitive load caused by factors outside of the instructional material, such as distractions, poor instructions, or irrelevant information. Germane cognitive load – This refers to cognitive load that is beneficial to learning, because it is related to the information in the current learning task and the learner's long-term memory.

Presenting the instructional material in an appropriate way according to different tasks can help learners understand more easily, and at the same time, efforts should be made to avoid generating intrinsic and extraneous cognitive loads. Using multimedia materials for teaching and practice can help learners reduce cognitive load. Currently, there is a large amount of educational research using cognitive load theory to explain the potential benefits of using multimedia to enhance learning. Multimedia can present information to students in a flexible way, involving various combinations such as text (written or spoken), static graphics (such as pictures, illustrations, graphics, and charts), dynamic graphics (such as animations) and videos. Learning with both text and pictures is more effective than learning with text or pictures alone (Butcher, 2014).

Currently, there are many examples of using multimedia to influence cognitive load. Çakiroğlu *et al.* used Scratch's visual image-based interface to stimulate learners' interest and reduce the cognitive load of novice programmers (Çakiroğlu *et al.*, 2018). Lavy used music as an intermediary to teach Scratch programming and further reduce the cognitive load of novice programmers (Lavy., 2023). Shim *et al.* introduced educational robots to stimulate student learning motivation and reduce the cognitive load of novice programmers (Shim *et al.*, 2016).

2.2. *CT Definition*

Wing (Wing, 2008) was the first to recognize CT as a problem-solving method. Further, in 2011, Wing defined CT as the thinking process in formulating a problem and its solution (Cuny *et al.*, 2010). The solution was expressed in a way that an information processing agent can effectively execute. After Wing proposed CT, other scholars put forward their opinions and provided relevant research. For example, in 2011, Tinker (National Research Council, 2011) believed that the core of CT is "the ability to decompose large problems into small problems until these small problems can be automatically solved to achieve rapid response." This definition has appeared frequently in subsequent research. To bring CT into K-12 education as soon as possible, the International Society for Technology in Education and the Computer Science Teachers Association have developed an operational definition of CT. Its problem-solving process includes problem formulation, data analysis, abstraction, modeling, and automation steps to achieve the most effective solution (Barr *et al.*, 2011).

Brennan and Resnick (Brennan and Resnick, 2012) proposed a CT framework comprising three dimensions, including CT concepts, CT practices, and CT perspectives, by analyzing Scratch. The following are some of the CT skills in each dimension:

(a) **CT concepts** include sequences, loops, parallelism, events, conditionals, operators, and data.

(b) **CT practices** include being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing.

(c) **CT perspectives**: expressing, connecting, and questioning can be transferred and applied to other programming environments.

Table 1

CT framework proposed by Brennan and Resnick (Brennan and Resnick, 2012)

CT skill	Definition
Concepts	
Sequences	A specific activity or task is represented as a series of individual steps or instructions that can be performed by a computer.
Loops	Loops are a mechanism for running the same sequence multiple times.
Parallelism	Parallelism refers to a sequence of instructions occurring simultaneously.
Events	One thing causes another to happen, which is an important element of interactive media.
Conditionals	The ability to make decisions based on specific conditions, which supports the expression of multiple outcomes.
Operators	Operators provide support for mathematical, logical, and string expressions, which enable programmers to perform numeric and string operations.
Data	Data involves storing, retrieving, and updating values.
Practices	
Incremental and iterative	An iterative process of gradually designing and implementing a solution.
Testing and debugging	Trial and error process to test and eliminate guaranteed failures.
Reusing and remixing	Build reusable instructions; develop new products based on the work of others.
Abstracting and modularizing	Build complex systems from basic elements.
Perspectives	
Expressing	Think of computing as a means of expression and creativity.
Connecting	Think of computing as a way to interact and work with other people.
Questioning	Ask questions and use technology to solve real-world problems.

Table 1 lists the skills and definitions included in the CT framework proposed by Brennan and Resnick (Brennan and Resnick, 2012).

2.3. *CT Assessment*

As with the diversity of definitions of CT, there is no widely accepted methodology to assess CT. As a result, multiple CT assessment techniques have been developed, making it difficult to measure the effectiveness of interventions in a reliable and valid manner (Valerie *et al.*, 2017). Questionnaire surveys, performance/combination assessments, knowledge and aptitude tests, and personal interviews are the four main types of CT assessments (Valerie *et al.*, 2017). Commonly used CT skill assessments focus on cognitive domains (Anderson and Krathwohl, 2001), such as creation and review using Dr. Scratch, analysis and application using Bebras, and comprehension and memory using CT test (CTt) (Cutumisu *et al.*, 2019).

Dr. Scratch is an automated tool for formative assessment of Scratch projects (Moreno-León *et al.*, 2016) that targets seven dimensions of CT capabilities: abstraction, parallelism, logical thinking, synchronization, flow control, user interactivity, and data representation. Each of these dimensions is measured on a scale of 0–3, and the

overall CT score is calculated by summing the partial scores. Dr. Scratch was designed to help teachers evaluate the Scratch projects of their students and detect common errors and poor programming habits. In addition, it can also provide statistics for organized programming teaching projects to evaluate their effectiveness (Moreno-León and Robles, 2015).

Computational thinking test (CTt) (González, 2015) is a popular assessment method for block-based programming languages because it is not restricted to a specific subject or programming language. CTt uses a multiple-choice question format, and it is one of the few validated assessment methods based on a practical guide to international standards for applied psychological and educational testing in secondary schools (Buffum *et al.*, 2015). CTt is designed to assess students aged 12–14 (7–8 grades). However, it can also be used for lower (5–6 grades) and higher grade students (9–10 grades). The test includes 28 items and takes approximately 45 min to complete. It emphasizes various computational concepts such as direction and sequences, loops, if conditions, and simple functions (Cutumisu *et al.*, 2019). These computational concepts are used to evaluate four cognitive processes in CT: decomposition, pattern recognition, abstraction, and algorithm design.

Bebras, which originated in Lithuania, is a CT assessment method that is independent of programming languages. Bebras does not require participants to write code; thus, using Bebras for assessment does not require participants to know a specific programming language. The Bebras Challenge on Informatics is conducted annually in more than 60 countries and regions worldwide (Bebras International Challenge on Informatics and Computational Thinking, 2023). The competition topic combines multiple challenges to promote short-term problem-solving skills related to informatics and CT for students of different ages (Dagienė and Stupurienė, 2016). To evaluate the impact of specific courses, Bebras primarily uses a programming foundation and programming knowledge to solve problems in real-world scenarios and evaluate the CT skills of the students (Hubwieser and Mühling, 2014).

For the competition, Bebras launches a new set of test questions each year, and these test questions are categorized as easy, medium, and hard based on difficulty. The participants were divided into 6 groups by age: Pre-Primary (1–2 grades, 5–8 years old), Primary (3–4 grades, 8–10 years old), Benjamins (5–6 grades, 10–12 years old), Cadets (7–8 grades, 12–14 years old), Juniors (9–10 grades, 14–16 years old), and Seniors (11–12 grades, 16–18 years old) (Bebras International Challenge on Informatics and Computational Thinking, 2023). There are approximately 12–15 questions in this test, and each question has a 3-min time limit. Thus, the total test time is approximately 36–45 minutes. The test questions are designed to be challenging, interesting, and thought-provoking for students.

The questions of Bebras are designed to increase the interest of students in information technology, thereby enhancing their motivation to learn. However, as it has grown in scale, it has attracted the attention of psychometric researchers who have begun to explore its potential as a measurement instrument for CT (Román-González *et al.*, 2017). Although Bebras was not designed specifically for CT testing, due to its wide

Table 2
Comparison of assessment methods

Assessment method	Programming experience	Applicable education level	Testing time	Language used
Dr. Scratch	Scratch	Elementary, middle, and High school	Not fixed	Multinational
CTt	Not limited	Middle and High School	45 minutes	Spanish and English
Bebras	Not limited	Elementary, middle, and High school	36–45 minutes	Multinational

age coverage and high degree of freedom in use, as well as its aim to seek "real" solutions, CT can be transferred and projected through the context of problems and used to solve problems for students (del Olmo-Muñoz *et al.*, 2020). Many scholars have used Bebras for experimental. For example, del Olmo-Muñoz *et al.* (del Olmo-Muñoz *et al.*, 2020) conducted a controlled experiment with second grade elementary school students using Code.org with one group using plugged-in devices and the other group using unplugged devices, and found that the CT skill level improved more significantly in the unplugged group. Chiazzese *et al.* (Chiazzese *et al.*, 2019) conducted a controlled experiment with elementary school students in grades 3–4 using Lego robot education kits as the experimental group and traditional curriculum as the control group. The total scores obtained by the experimental group were significantly higher than those of the control group. Chen *et al.* (Chen *et al.*, 2018) conducted a controlled experiment in which high school students in the second grade were taught Arduino using a situational learning strategy, and were compared to students who were taught using traditional learning methods. The results showed that students who used the situational learning strategy had better computational thinking abilities than those who used traditional learning methods.

The comparison of the above three assessment methods is shown in Table 2. Bebras is used in this study to investigate changes in students' CT after considering the applicability and completeness of elementary school education.

2.4. *Self-efficacy Scale*

Self-efficacy refers to the strength of a person's belief in his/her ability to master or complete a task (Bandura, 1994), and it can also be used to assess a learner's perception of his/her CT skills. Self-efficacy is a critical driver of a person's perseverance and resilience when confronted with adversity (Nordén *et al.*, 2017). Students with high self-efficacy see task completion as a challenge to be solved rather than something to be avoided. Self-efficacy affects the effort exhibited in various learning situations (Gandhi and Varma, 2010). In the programming context, self-efficacy is described as "a person's potency toward organizing and executing action processes to achieve a specified performance" (Kong, 2017).

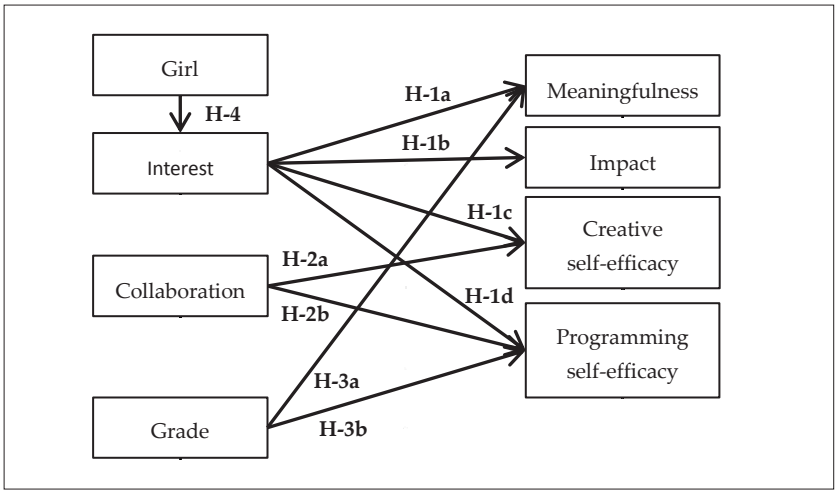


Fig. 1 Programming empowerment research framework (Kong *et al.*, 2018).

Based on Brennan and Resnick's Three-dimensional CT framework, Kong (Kong *et al.*, 2018) developed a programming empowerment scale. They believed that most current assessment methods only evaluated CT concepts and practices. Therefore, to evaluate more CT perspectives, the impact of investigation interest and cooperation based on self-efficacy were included in Kong's programming empowerment scale, and its corresponding structure is shown in Fig. 1.

Kong conducted the corresponding survey with a sample of 287 elementary school students in fourth to sixth grades. They found that interested students rated programming as meaningful and had higher self-efficacy. Students with a more positive cooperative attitude than others had higher creative self-efficacy, and boys showed more interest in programming than girls. Older students rated programming as less rewarding and had lower programming self-efficacy than younger students. Future research on programming can be conducted in an environment that provides students with collaboration opportunities while also exploring the impact of students' interests on both CT and programming courses.

3. OwlSpace Programming Course

3.1. *OwlSpace*

OwlSpace is a programming platform that combines programming, maker, information technology, and artificial intelligence education. OwlSpace integrates "teaching," "learning," and "practice" in a single platform to meet all the needs of students from entry to independent creation. This educational programming language is easy to teach,

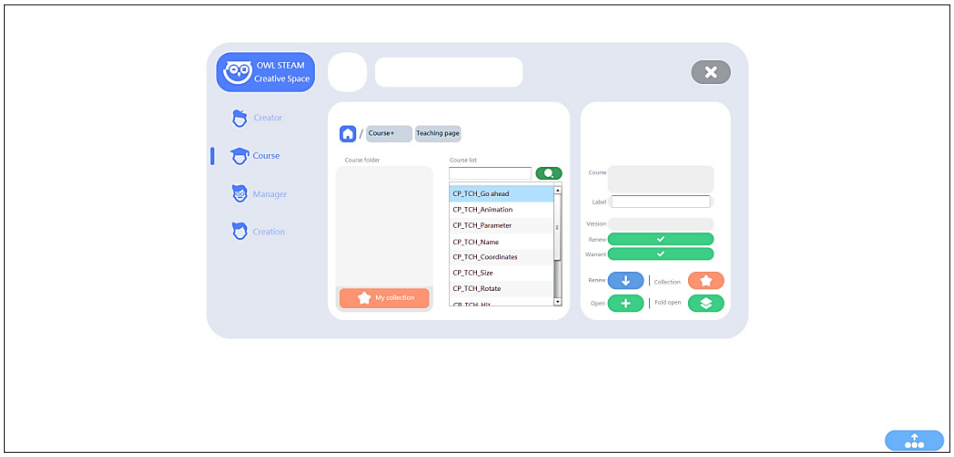


Fig. 2 OwlSpace course selection interface.

learn, and use. It employs graphics and animation to illustrate the program's operation so that students are not overly challenged or bored while learning it. In addition, to help beginner students who transition to mainstream programming languages, it combines the characteristics of mainstream programming languages, such as C++ and JavaScript. Furthermore, students can quickly develop IoT devices and automated robots by connecting to micro:bit, Arduino, ESP32, and other microprocessors.

OwlSpace has a built-in series of fundamental to advanced programming courses, with the level of difficulty scaled up to accommodate elementary, middle, and high school users. Thus, students learn joint programming statements such as variables, conditions, loops, and functions. OwlSpace simultaneously trains the students' CT skills by arranging exercises for them to perform after courses. The programming course selection interface is shown in Fig. 2.

3.2. *OwlSpace Programming Language*

Compared to block-based programming languages that drag blocks to build programs, OwlSpace programming language does not utilize blocks. Instead, it requires the user to input the name and properties of the object, e.g., bee's X coordinates, as variables. It sets the specified value as the object's value, using the equal sign to establish the basic code. Fig. 3 shows the bee's positional movement and size change through code.

For conditional expressions, the user must input three strings: "if," "else," and "end" as the structure. For example, "bee.height > butterfly.height" in brackets is the selection condition used to determine whether the bee or butterfly moves to the flower. Depending on the selection condition "is true or not," the bee can move to the flower using "bee.moveTo(flower)" or the butterfly can move to the flower using "butterfly.moveTo(flower)." Fig. 4 shows the corresponding code and illustrates the execution results.

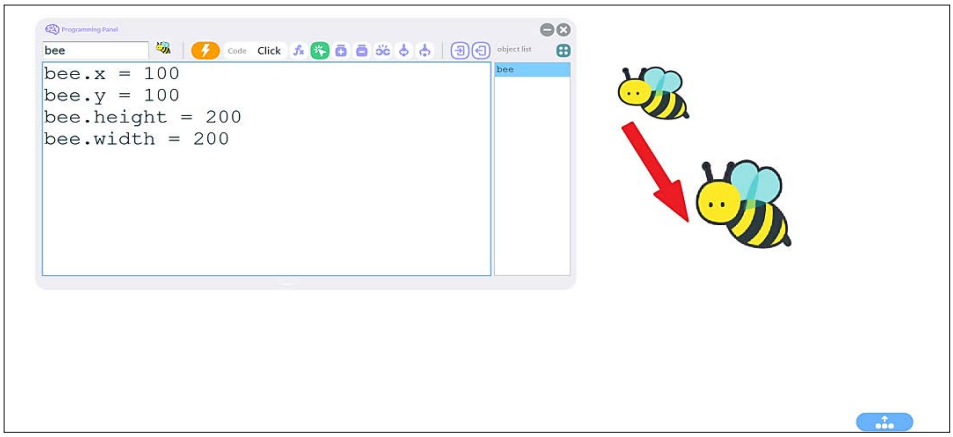


Fig. 3 Code to adjust the coordinates, height, and width of the bee.

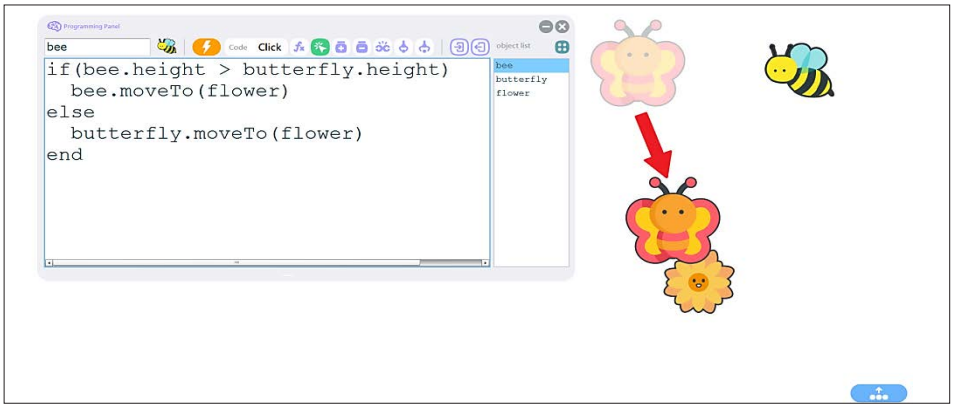


Fig. 4 Conditional statement to select a mobile bee or butterfly.

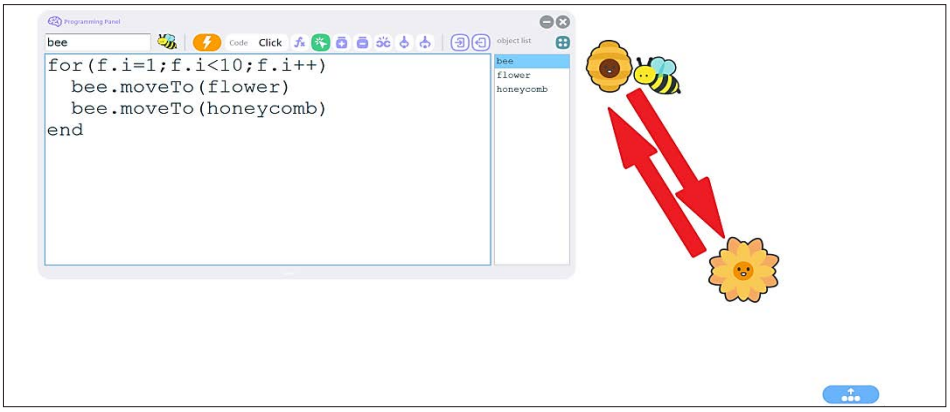


Fig. 5 Loop statement to move the bee repeatedly.

Another joint basic program is a loop, which executes the "For loop" operation through "for" and "end" strings. Here, the user enters the code that needs to be executed repeatedly in the area between "for" and "end," and the program can be executed continuously according to the set number of loops. Fig. 5 shows how the bee repeatedly moves between the flower and the honeycomb.

3.3. *Effect of OwlSpace Programming Courses on CT*

OwlSpace platform provides built-in instructional courses and practice exercises, where students can follow on-screen instructions to start coding programs. The course content includes object naming, coordinate settings, length and width settings, object movement, rotation, collision, as well as learning how to use functions, variables, expressions, and conditional expressions. To reduce students' cognitive load, each course uses vivid pictures and animation to illustrate the learning objectives and allows students to think of how to use the functions they learned when they practice after class. The instructional screen is shown in Fig. 6.

The students input code according to the requirements of the task to complete their creation. They make the pufferfish bigger than the shark, but smaller than the chef, and have the shark and chef say the corresponding sentences when the conditions are not met. During the process, students can discuss with others or complete the task on their own. When the students complete the task correctly, the word "YES!" will be displayed on the screen, as shown in Fig. 7 and Fig. 8.

This programming course includes a total of 11 units with the theme of basic programming. The corresponding table of each learning content (unit) is planned prior to teaching to ensure that the course's content covers the Three-dimensional CT framework, as shown in Table 3.

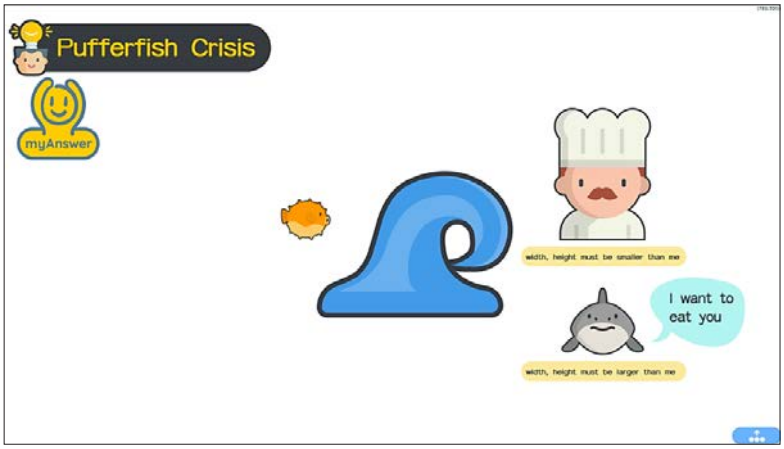


Fig. 6 Built-in teaching of OwlSpace.

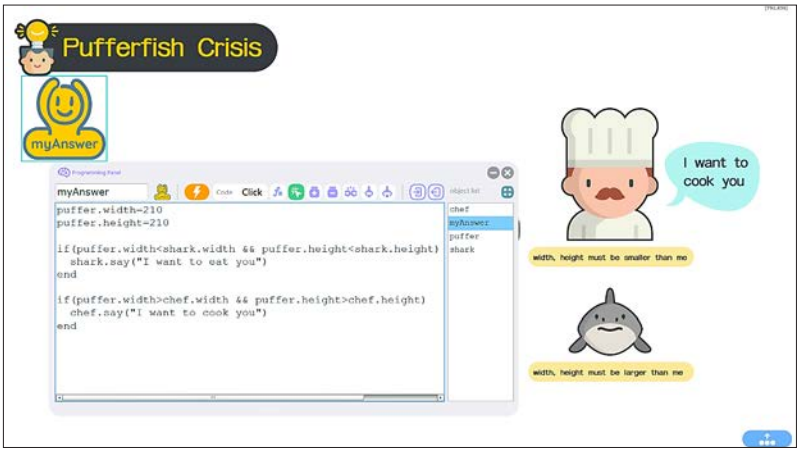


Fig. 7. Student uses programs to answer questions.

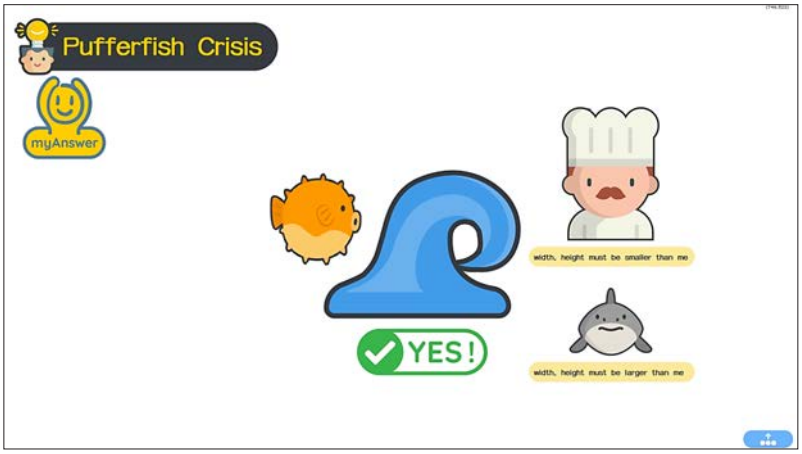


Fig. 8. The student answered correctly.

3.4. *OwlSpace Programming Research Process*

In this study, we investigated the relationship between changes in students' CT and different background variables on CT learning after the students went through the proposed curriculum in OwlSpace programming platform. The participants were students in fourth and fifth grades at an elementary school in Taiwan. The research involved interventions. Thus, pre- and post-assessment designs were used. In addition, two assessment tools were utilized to realize comprehensive assessment, i.e., the previous Bebras competition test questions and programming empowerment scale proposed by Kong *et al.* (Kong *et al.*, 2018) This study also conformed to the Three-dimensional CT framework (concepts, practices, and perspectives) proposed by Brennan and Resnick (Brennan and Resnick, 2012). In addition, relevant background data (gender, grade, ex-

Table 3

OwlSpace programming course CT definition correspondence table

Unit	Concepts							Practices				Perspectives		
	Sequences	Loops	Parallelism	Events	Conditionals	Operators	Data	Being incremental and iterative	Testing and debugging	Reusing and remixing	Abstracting and modularizing	Expressing	Connecting	Questioning
Name							✓		✓			✓		
Coordinates	✓					✓	✓	✓			✓	✓	✓	
Size	✓				✓	✓	✓		✓			✓	✓	✓
Scale	✓			✓	✓	✓	✓	✓	✓			✓		
Function	✓	✓		✓						✓	✓	✓		✓
Parameter	✓	✓		✓			✓	✓	✓	✓	✓	✓	✓	
Alpha	✓			✓	✓		✓		✓			✓		✓
Animation	✓		✓	✓			✓	✓			✓	✓		✓
Go ahead	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓		
Rotate	✓	✓		✓		✓	✓		✓	✓		✓	✓	✓
Hit	✓		✓	✓	✓	✓	✓	✓			✓	✓	✓	

perience, etc.) were collected for subsequent analysis. Table 4 shows the experimental design. Bebras serves as the pre-test; the intervention step is teaching OwlSpace programming course; Bebras is repeated as post-test; and finally, programming empowerment scale is implemented.

As the participants were elementary school students, we began by selecting suitable questions from the Bebras problem set based on the elementary school level. Next, we utilized Dagienė's (Dagienė *et al.*, 2017) proposed two-dimensional classification system and classified the Bebras questions using the provided keywords for each question. Subsequently, we specifically chose questions falling under the categories of "Algorithms and Programming" and "Data, Data Structures, and Representations." Finally, we correlated the filtered questions with CT skills to ensure that students could apply their learned CT skills to solve the questions. Table 5 presents the relationship between the selected questions and CT skills. In addition, we limited the number of questions to ten (Appendix I) to avoid disrupting the course's progress.

Table 4

Teaching experiment design

Experimental design	Pre-test	Intervene	Post-test
One-group pre-post-test design	O_1	*	O_2 and O_3

O_1：Bebras test (pre-test)
*：Program teaching
O_2：Bebras test (post-test)
O_3：Programming empowerment scale

Table 5
A table for task categorization using the two-dimensional categorization system

Name of task	Informatics domain	Keywords	CT skill
1. Bracelet	Data, data structures and representations	Recognising patterns Grammar check	Abstracting and modularizing Reusing and remixing
2. Dog swap	Algorithms and programming	Sorting algorithm Bubble sort	Data Incremental and iterative
3. Lollipop and toothbrush	Algorithms and programming	Constraints Best solution	Operators Incremental and iterative
4. Spinning squares	Algorithms and programming	Divide and conquer Sorting Track	Parallelism Abstracting and modularizing
5. Painting robot	Data, data structures and representations	Graph theory Eulerian path	Incremental and iterative
6. Board game piece returns home	Algorithms and programming	Program execution	Sequences
7. Bebras painting	Algorithms and programming	Algorithm Computer vision Image conversion	Events Abstracting and modularizing
8. Dancer	Algorithms and programming	Brute force algorithm Binary	Conditionals Abstracting and modularizing
9. Secret recipe	Data, data structures and representations	Linked list	Data Abstracting and modularizing Reusing and remixing
10. Bakery	Algorithms and programming	Loop algorithm Operating systems Scheduling	Loops Abstracting and modularizing Incremental and iterative

Prior to conducting the experiment, we consulted with the computer science teacher and confirmed that the students at the school had not previously participated in the Bebras competition. Hence, it can be assumed that the students had no prior exposure to any Bebras questions. To maintain the integrity of the study, we explicitly requested the teachers not to reveal that the questions were sourced from the Bebras competition to prevent students from searching for related information online. Following the completion of the pre-test, we refrained from providing the correct answers to the students to prevent them from memorizing the answers. After teaching the programming course, the post-test covered the same topics as the pre-test. Furthermore, using a programming empowerment scale, we investigated whether the students' assumptions about the course's ability to contribute to creativity, cooperative attitudes, and programming self-efficacy were met. The questions of programming empowerment scale are shown in Table 6, and each question is investigated using a 5-point Likert scale.

Table 6
The questions of programming empowerment scale

Construct	Questions	source
Meaningfulness	Programming is useful to me. Programming will help me achieve my goals. I want to become good at programming. Programming is important to me.	(Kong *et al.*, 2018)
Impact	I want to use programming to help solve problems in the world. I want to use programming to improve people's lives. I can use programming to make daily life easier.	(Kong *et al.*, 2018)
Creative self-efficacy	I would like to design things using programming. Computer programmers are creative. It is important to be creative when you are programming.	(Kong *et al.*, 2018)
Programming self-efficacy	I can learn how to program. I am good at programming. I think of myself as someone who can program. I have the skills to program. I have confidence in my ability to program.	(Kong *et al.*, 2018)
Interest in programming	Programming is interesting. I am curious about the content of programming. I think the content of programming is fun. I am very interested in computer programming activities.	(Kong *et al.*, 2018)
Attitude toward collaboration in programming	I like to program with others. I finish things faster when I program with others. I have better ideas when I program with others. People ask me for help with computers a lot.	(Kong *et al.*, 2018)

3.5. *Framework Used in this Study*

We utilized Kong *et al.*'s programming empowerment scale to establish a research framework, as shown in Fig. 9, and established 9 hypotheses (H-1a, H-1b, H-1c, H-1d, H-2a, H-2b, H-3a, H-3b, and H-4) that were identical to Kong *et al.*'s research.

Kong *et al.* believes that CT courses are novel to students and can stimulate their interest (Kong *et al.*, 2018). When students are interested in programming, it will have greater significance and impact on them. They will be willing to spend more time and effort to challenge difficult tasks, thereby having higher levels of creative self-efficacy and programming self-efficacy.

- H-1a. Students with greater interest in OwlSpace programming than others view it as more meaningfulness.
- H-1b. Students with greater interest in OwlSpace programming than others believe it has greater impact.
- H-1c. Students with greater interest in OwlSpace programming than others have greater creative self-efficacy.

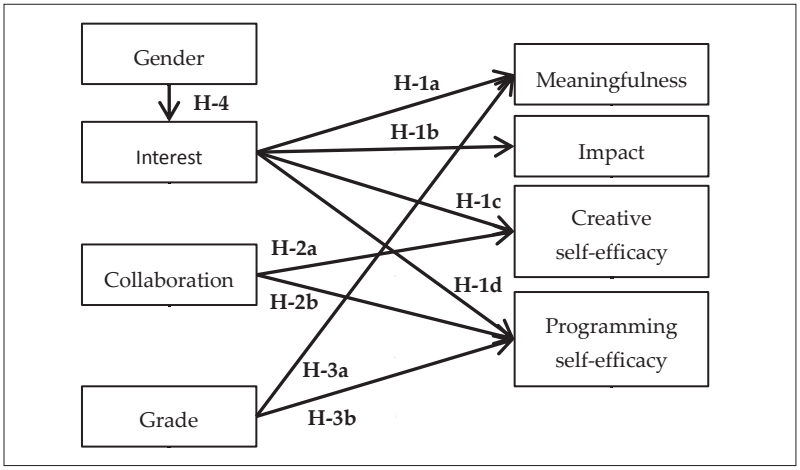


Fig. 9. Programming empowerment research framework.

- H-1d. Students with greater interest in OwlSpace programming than others have greater programming self-efficacy.

Kong *et al.* believed that collaborative attitude is related to creative self-efficacy and programming self-efficacy (Kong *et al.*, 2018). Students with a collaborative attitude tend to work harder and solve difficult tasks more effectively when working with others, resulting in better programming abilities.

- H-2a. Students with better attitudes toward collaborative programming than others have greater creative self-efficacy.
- H-2b. Students with better attitudes toward collaborative programming than others have greater programming self-efficacy.

Kong *et al.* reviewed previous research findings and found that older students usually perceive school subjects as meaningless compared to younger students (Kong *et al.*, 2018). This is because older students usually face more disconnected teaching situations that are not related to their daily lives, which reduces their motivation to learn. Therefore, they tend to believe that computer programming is not meaningful and believe that their programming skills are weaker.

- H-3a. Students of different grades think that the meaning of OwlSpace programming is different.
- H-3b. Students of different grades have different levels of self-efficacy in OwlSpace programming.

Kong *et al.* reviewed previous studies and found that boys are generally more interested in computer programming, and gender stereotypes of male computer experts can hinder girls' interest in computer programming (Kong *et al.*, 2018). Therefore, it is believed that boys are usually more interested in computer programming than girls.

- H-4. Students of different genders have different levels of interests in OwlSpace programming.

To understand the effectiveness of students using OwlSpace courses, Statistical Product and Service Solutions (SPSS) statistical package software was used to analyze the Bebras test results. Here, a paired sample T-test and an independent sample T-test were conducted using SPSS to explore the differences between the students' learning effects and their gender, grade, and past programming experience. Programming empowerment questionnaire was analyzed and validated using SPSS regression analysis of hypotheses H-1a to H-2b. H-3a to H-4 were examined for differences based on gender and grade level using independent sample T-tests.

4. Results

4.1. *Descriptive Statistics*

This experiment started in February 2022 and ran until July 2022. Table 7 shows the descriptive statistical data from the questionnaire. The participants were 150 elementary school students in the fourth and fifth grades. The programming language learned on programming empowerment scale is a multiple-answers question.

A total of 150 questionnaires were returned. We omitted 12 invalid (incomplete) questionnaires. Thus, the total number of valid questionnaires was 138. Table 7 shows that the proportion of male students is higher than that of female students. In addition, fifth grade students make up the majority of participants; 52.9% of the students had no programming experience, and 37.7% of students had learned some programming using Scratch.

Table 7
Student background statistics

Attribute	People	Percent
Male students	80	58.0%
Female students	58	42.0%
Fourth grade	58	42.0%
Fifth grade	80	58.0%
Programming experience	65	47.1%
No programming experience	73	52.9%
Scratch	52	37.7%
Code.org	0	0%
Alice	1	0.7%
Java	2	1.4%
Python	7	5.1%
Other	15	10.9%

4.2. *CT Skills Before and After Test Analysis.*

Table 8 shows that the average pre- and post-test scores were 48.91 and 53.04, respectively; thus, the average post-test score was 4.13 points higher than the pre-test score. The paired samples T-test was conducted to compare the two test scores and determine whether the pre-test and post-test scores were statistically different. Here, the T value was −2.064, and the p-value was 0.041. These results indicate a significant difference and demonstrate that students made progress on the CT.

We found that female students scored 48.97 in the pre-test on average, while male students scored 48.88. Thus, the female students scored 0.09 points higher than the male students, and the female students scored 1.89 points higher than the boys in the post-test. However, the results of the independent sample T-test comparison indicate that the statistical difference between the two is negligible. For both the pre-test and post-test, no significant difference was observed in terms of gender affecting CT. The results are shown in Table 9.

For the pre-test, the scores obtained by the students in the fourth and fifth grades were 47.76 and 49.75, respectively. The scores obtained by fifth grade students are higher than those in fourth grade by 1.99 points on the pre-test and 4.66 points on the post-test. No statistical difference between the pre-test and the post-test results were found using the independent sample T-test comparison, and the students' grade did not affect the CT. The results are shown in Table 10.

Students with prior programming experience performed better on the test than those without similar experience. However, the independent sample T-test comparison did not reveal a significant difference. The results are shown in Table 11.

Table 8
Paired sample T-test of pre-test and post-test.

N=138	Mean (Standard Deviation)		df	T	P
	Pre-test	Post-test			
Score	48.91 (21.36)	53.04 (20.98)	137	−2.064	0.041

Table 9
Independent sample T-test based on student's gender

Test	Male students	Female students	T	P
Pre-test	48.88	48.97	-0.024	0.981
Post-test	52.25	54.14	-0.520	0.604

Table 10
Independent sample T-test by grades

Test	Fourth grade	Fifth grade	T	P
Pre-test	47.76	49.75	-0.539	0.591
Post-test	50.34	55.00	-1.290	0.199

Table 11
Independent sample T-test of past programming experience

Test	Experience	No experience	T	P
Pre-test	52.46	45.75	1.857	0.065
Post-test	52.62	53.42	-0.225	0.822

4.3. *CT Self-efficacy Questionnaire Analysis*

The questionarire is shown in Appendix II. Programming empowerment scale questionnaire uses a 5-point Likert scale to investigate level of self-efficacy of users. As shown in Table 12, the average value of each item is higher than 3, indicating a middle-to-high level of self-efficacy. In addition, Cronbach's alpha reliability is greater than 0.7, which indicates good reliability.

Table 12
Programming enablement statistics

Item	Mean	SD	Cronbach's alpha
Meaningfulness 1	4.15	0.734	0.899
Meaningfulness 2	4.00	0.755	
Meaningfulness 3	4.07	0.794	
Meaningfulness 4	3.88	0.793	
Impact 1	3.96	0.858	0.827
Impact 2	4.09	0.782	
Impact 3	4.07	0.834	
Creative self-efficacy 1	4.25	0.695	0.853
Creative self-efficacy 2	4.38	0.642	
Creative self-efficacy 3	4.42	0.626	
Programming self-efficacy 1	4.25	0.715	0.912
Programming self-efficacy 2	3.50	0.930	
Programming self-efficacy 3	3.57	0.920	
Programming self-efficacy 4	3.59	0.851	
Programming self-efficacy 5	3.62	0.890	

Continued on next page

Table 12 – continued from previous page

Item	Mean	SD	Cronbach's alpha
Interest 1	4.27	0.700	0.931
Interest 2	4.17	0.819	
Interest 3	4.23	0.708	
Interest 4	4.09	0.782	
Collaboration 1	4.08	0.855	0.822
Collaboration 2	4.01	0.819	
Collaboration 3	3.97	0.837	
Collaboration 4	3.38	1.028	

Table 13 and Fig. 10 show regression analysis values of different factors. The data allows us to identify the following observations.

- When using linear regression to analyze the effect of interest on meaningfulness, the significance p-value < 0.05. Thus, hypothesis H-1a, i.e., students who are more interested in programming think it is more meaningful, is valid.
- When using linear regression to analyze the effect of interest on impact, the significance p-value < 0.05. Thus, hypothesis H-1b is valid, i.e., students who are more interested in programming think it has a greater impact.
- When using linear regression to analyze the influence of interest on creative self-efficacy, the significance p-value < 0.05. Thus, it is assumed that H-1c is valid, i.e., students who are more interested in programming have greater creative self-efficacy.
- When using linear regression to analyze the influence of interest on programming self-efficacy, the significance p-value < 0.05. Thus, it is assumed that hypothesis H-1d is confirmed, i.e., students who are more interested in programming have higher programming self-efficacy.
- When using linear regression to analyze the influence of collaborative programming attitude on creative self-efficacy, the significance p-value < 0.05. Thus, it is assumed that H-2a is confirmed, i.e., students with a better collaborative programming attitude have greater creative self-efficacy.

Table 13
Regression analysis results

Hypothesis	Independent Variable	Dependent Variable	Beta	T	P
H-1a	Interest	Meaningfulness	0.749	13.19	0.000
H-1b	Interest	Impact	0.712	11.84	0.000
H-1c	Interest	Creative self-efficacy	0.731	12.50	0.000
H-1d	Interest	Programming self-efficacy	0.649	9.94	0.000
H-2a	Collaboration	Creative self-efficacy	0.506	6.84	0.000
H-2b	Collaboration	Programming self-efficacy	0.615	9.09	0.000

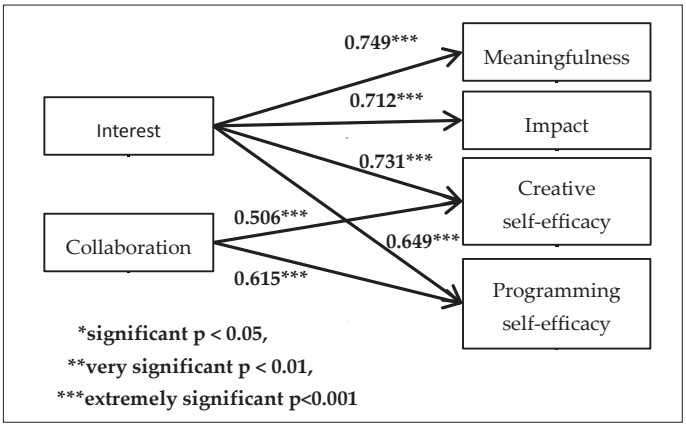


Fig. 10 Programming empowerment path relationship diagram.

- When using linear regression to analyze the influence of collaboration on programming self-efficacy, the significance p-value < 0.05. Thus, H-2b is valid, i.e., students with better attitudes toward collaborative programming have higher programming self-efficacy.

Table 14 shows the difference analysis between different grades, from which we conclude the following.

- Using the independent sample T-test to compare the mean of the fourth and fifth grades, the significant p-value in meaningfulness and programming self-efficacy was not less than 0.05. Thus, H-3a and H-3b are rejected, and the means of fourth and fifth grades did not differ. Therefore, we cannot assume that students in different grades think that the meaning of OwlSpace programming course is different (H-3a). In addition, we cannot assume that students of different grades have different programming self-efficacy when using OwlSpace (H-3b).

Table 15 shows the results of analyzing the differences between male students and female students. Using the independent sample T-test to compare the differences between the average numbers of male students and female students, the significant p-value in interest was not less than 0.05. Thus, H-4 is rejected, i.e., the average numbers of male students and female students do not differ. Therefore, assuming that H-4 students of different genders have different interests in OwlSpace programming course is invalid.

Table 14
Independent samples T-test of grades

No.	Grade				T-test results for Interested	
	Fourth grade		Fifth grade			
	Mean	SD	Mean	SD	T	P
Meaningfulness	4.01	0.659	4.03	0.688	-.184	0.854
Programming	3.62	0.705	3.77	0.769	-1.165	0.246

Table 15
Independent samples T-test of gender

No.	Gender				T-test results for Interested	
	Male students		Female students			
	Mean	SD	Mean	SD	T	P
Interest	4.24	0.688	4.12	0.682	0.978	0.330

Table 16
Hypothesis verification results

Hypothesis	Hypothetical content	Result
H-1a	Students with greater interest in OwlSpace programming than others view it as more meaningfulness.	Valid
H-1b	Students with greater interest in OwlSpace programming than others believe it has greater impact.	Valid
H-1c	Students with greater interest in OwlSpace programming than others have greater creative self-efficacy.	Valid
H-1d	Students with greater interest in OwlSpace programming than others have greater programming self-efficacy.	Valid
H-2a	Students with better attitudes toward collaborative programming than others have greater creative self-efficacy.	Valid
H-2b	Students with better attitudes toward collaborative programming than others have greater programming self-efficacy.	Valid
H-3a	Students of different grades think that the meaning of OwlSpace programming is different.	Invalid
H-3b	Students of different grades have different levels of self-efficacy in OwlSpace programming.	Invalid
H-4	Students of different genders have different levels of interests in OwlSpace programming.	Invalid

The final hypothesis verification results are shown in Table 16. A total of six assumptions are confirmed, and three assumptions are not confirmed. OwlSpace programming language courses do not have grade and gender differences. Interested students find OwlSpace meaningful and influential, and students that are interested or those with better attitudes toward collaborative programming exhibit greater creative self-efficacy.

5. Conclusions

In this study, we utilized the Bebras test and programming empowerment scale combined with Brennan's Three-dimensional CT framework to investigate whether elementary school students could improve their CT skills after training with OwlSpace programming courses. Pre- and post-test data analysis show that students improved on the test, and all items on programming empowerment scale are positive. Therefore,

we believe that by using OwlSpace programming courses, students can cultivate and improve the CT concepts, practices, and perspectives. Gender, grade, and past experience are frequently discussed as factors that cause differences in students' learning of programming. However, this was not observed in this study. This result is consistent with Papadakis *et al.*'s study(Papadakis *et al.*, 2016), which found that gender does not affect children's performance in computational thinking and digital skills in early education. According to Angeli and Valadines (Angeli and Valanides, 2020), it is not because girls have lower programming and computational thinking skills or motivation levels, but rather because of a lack of creative content that meets the needs and interests of children. Girls can also perform well as long as they are provided with appropriate teaching environments.

In terms of programming empowerment scale model, we found that six corresponding assumptions hold true. Students who expressed greater interest in programming with OwlSpace perceived it as more meaningful, as having a greater impact, as having more creative self-efficacy, and as having higher programming self-efficacy, which is consistent with Kong's findings. Students with better attitudes toward collaboration exhibited higher creative self-efficacy and higher programming self-efficacy, which indicates that thay possess both "expressing" and "connecting" from CT perspectives. In addition, differences in grade and gender had no effect on students' meaning, interest, and self-efficacy in programming, which contradicts the findings of Kong's study. However, these findings are consistent with the research of Ma *et al.*(Ma *et al.*, 2021) Using problem-solving instruction can enhance the CT self-efficacy of older students and girls. Additionally, providing supportive instruction can increase girls' engagement and confidence in using programming environments.

OwlSpace platform used in this study provides programming courses suitable for each learning stage according to the students' academic background. For example, basic courses for elementary school, advanced courses for junior high school, and professional courses for high school, with assisted teaching provided to help students make a gradual transition in their learning. Ultimately, students can independently use formal text-based programming languages such as C or Python. However, in this study, we only investigated whether students improved their CT skills by taking OwlSpace courses. We did not compare OwlSpace interactively with other joint programming tools. Therefore, we intend to conduct controlled experiments in the same environment in the future to investigate differences in the students' use of various programming education systems and platforms.

Acknowledgement

This work was partly supported by the Ministry of Education, Taiwan, Republic of China, under the "Enhancing Practical Competencies of Teachers and Students through the Development of Digital Teaching Materials for the Introduction of Text-based Programming Language, Neural Networks, and Information Hiding Concepts" Project.

References

Anderson, L.W., Krathwohl, D.R. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York.

Angeli, C., Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction efect between gender and scafolding strategy. *Computers in Human Behavior*, 105, 105954.

Bandura, A. (1994). Social cognitive theory and exercise of control over HIV infection, In Preventing AIDS, pp. 25–59. Springer, Boston, MA.

Barr, D., Harrison, J., Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, 38(6), 20–23.

Bebras International Challenge on Informatics and Computational Thinking (2023). Participation in international Bebras challenges worldwide. Retrieved 24 July 2023. Accessed at `https://www.bebras.org/statistics.html`

Bebras International Challenge on Informatics and Computational Thinking (2023). Structure of a challenge. Retrieved 24 July 2023. Accessed at `https://www.bebras.org/structure.html`

Brennan, K., Resnick, M., 2012, 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, Vol. 1, p. 25.

Buffum, P.S., Lobene, E.V., Frankosky, M.H., Boyer, K.E., Wiebe, E.N., Lester, J.C. (2015). A practical guide to developing and validating computer science knowledge assessments with application to middle school. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pp. 622–627

Butcher, K.R. (2014). The Multimedia Principle. In: R.E.E. Mayer (Ed.), *The Cambridge Handbook of Multimedia Learning* (2nd ed.), pp. 174–205.

Çakiroğlu, Ü., Suiçmez, S.S., Kurtoğlu, Y.B., Sari, A., Yildiz, S., Öztürk, M. (2018). Exploring perceived cognitive load in learning programming via Scratch. *Research in Learning Technology*, Vol. 26.

Chen, J.M., Wu, T.T., Sandnes, F.E. (2018). Exploration of computational thinking based on bebras performance in webduino programming by high school students. In: *Innovative Technologies and Learning: First International Conference, ICITL 2018, Portoroz, Slovenia, August 27–30, 2018, Proceedings 1*, pp. 443–452.

Chetty, J., Barlow-Jones, G. (2012). Bridging the gap: The role of mediated transfer for computer programming. In: *Proceedings of Computer Science & Information Technology*, Vol. 43.

Chiazzese, G., Arrigo, M., Chifari, A., Lonati, V., Tosto, C. (2019). Educational robotics in primary school: Measuring the development of computational thinking skills with the Bebras tasks. *In Informatics*, 6(4).

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, 15(5), 107–116.

Cuny, J., Snyder, L., Wing, J. (2010). Demystifying computational thinking for non-computer scientists. *Educational Research Review*, 22(1).

Cutumisu, M., Adams, C., Lu, C. (2019). A scoping review of empirical research on recent computational thinking assessments. *Journal of Science Education and Technology*, 28(6), 651–676.

Cutumisu, M., Adams, C., Lu, C. (2019). A scoping review of empirical research on recent computational thinking assessments. *Journal of Science Education and Technology*, 28(6), 651–676.

Cynthia, S., Woollard, J. (2014). Computational thinking: the developing definition, In: *Proceedings of Conference: Special Interest Group on Computer Science Education (SIGCSE)*.

Dagienė, V., Sentance, S., Stupurienė, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, 28(1), 23–44.

Dagienė, V., Stupurienė, G. (2016). Bebras – a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 5(1), 25–44.

Gandhi, H. and Varma, M. (2010). Strategic content learning approach to promote self-regulated learning in mathematics. In: *Poceedings of epiSTME*, Vol. 3, 119–124.

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

Hubwieser, P., Mühling, A. (2014). Playing PISA with Bebras, In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, pp. 128–129.

Kong, S.C. (2017). Development and validation of a programming self-efficacy scale for senior primary school learners. In: *Proceedings of the International Conference on Computational Thinking Education*, pp. 97–102.

Kong, S.C., Chiu, M.M., Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers and Education*, 127, 178–189.

Lavy, I. (2023). Leveraging the Pied Piper Effect – The Case of Teaching Programming to Sixth-grade Students Via Music. *Informatics in Education*, 22(1), 45–69.

Lindberg, S.N., , Laine, T.H., Haaranen, L. (2018). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British Journal of Educational Technology*, 50(4), 1979–1995.

Ma, H., Zhao, M., Wang, H., Wan, X., Cavanaugh, T.W., Liu, J. (2021). Promoting pupils' computational thinking skills and self-efficacy: A problem-solving instructional approach. *Educational Technology Research and Development*, 69(3), 1599–1616.

van Merri€enboer, J.J.G., Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review*, 17(2), 147–177.

Moreno-León, J. (2018). On computational thinking as a universal skill, In: *Proceedings of 2018 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1684–1689.

Moreno-León, J., Robles, G. (2015). Dr. Scratch: A web tool to automatically evaluate Scratch projects. In: *Proceedings of the Workshop in Primary and Secondary Computing Education*, pp. 132–133.

Moreno-León, J., Robles, G., Román-González, M. (2016). Comparing computational thinking development assessment scores with software complexity metrics. In: *Proceedings of 2016 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1040–1045.

National Research Council (2011). *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. Washington, DC: National Academies Press.

Nordén, L.Å., Mannila, L., Pears, A. (2017). Development of a self-efficacy scale for digital competences in schools. In: *Proceedings of 2017 IEEE Frontiers in Education Conference (FIE)* , pp. 1–7.

del Olmo-Muñoz, J., Cózar-Gutiérrez, R., González-Calero, J.A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education*, 150, 103832.

Paas, F., Renkl, A., Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. *Educational Psychologist*, 38(1), 1–4.

Paas, F., Renkl, A., Sweller, J. (2004). Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. *Instructional Science*, 32(1/2), 1–8.

Paas, F.G., Van Merriënboer, J.J. (1994). Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive-load approach. *Journal of educational psychology*, 86(1), 122–133.

Papadakis, S., Kalogiannakis, M., Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: A case study. *International Journal of Mobile Learning and Organisation*, 10(3), 187–202.

Parsons, D., Haden, P. (2007). Programming Osmosis: Knowledge transfer from imperative to visual programming environments. In: *Procedings of the Twentieth Annual NACCQ Conference*, Vol. 209. New Zealand: Hamilton, p. 215.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Evelyn, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(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.

Shim, J., Kwon, D., Lee, W. (2016) The effects of a robot game environment on computer programming education for elementary school students. *IEEE Transactions on Education*, 60(2), 164–172.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2), 257–285.

Sweller, J., Ayres, P., Kalyuga, S. (2011). *Cognitive Load Theory*. New York, NY: Springer.

Valerie, J.S., Chen, S., Asbell-Clarke, J. (2017). Demystifying comptational thinking. *Educational Research Review*, 22, 142–158.

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

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

Wing, J.M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725.

Wolber, D., Harold A., Friedman, M. (2015). Democratizing computing with App Inventor, GetMobile: Mobile Computing and Communications, 18(4), 53–58.

W.-Y. Li is a PhD student in the Department of Information Management at Chaoyang University of Technology in Taichung, Taiwan. He earned his M.S.I.M degrees in information management from Chaoyang University of Technology in Taiwan in 2011 and 2013. His research focuses on program education, instructional software, and the design and development of human information systems

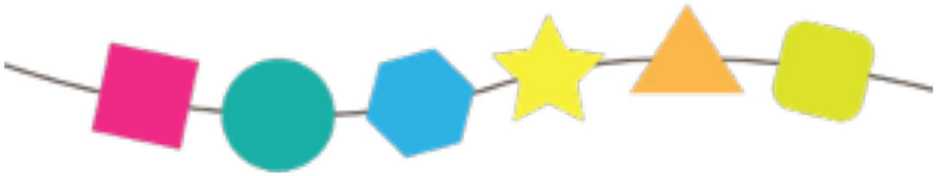
T.-C. Lu is Professor at Department of Information Management, Chaoyang University of Technology Taichung, Taiwan. Tzu-Chuen Lu received the B.M. and M.S.I.M. degrees in information management from the Chaoyang University of Technology, Taiwan, 1999 and 2001, respectively, and the Ph.D. degree in computer engineering from National Chung Cheng University, in 2006. Her research areas include computational complexity, steganography, data encapsulation, and decision support systems.

Appendix I

+ Computational thinking multiple-choice questions

1. Bracelet

Emily has broken her favorite bracelet. The broken bracelet now looks like this:



Question:
Which of the following four bracelets shows what the bracelet looked like when it was whole?

(A) (B) (C) (D)

2. Dog swap

Two different types of dogs are lined up in the following positions.

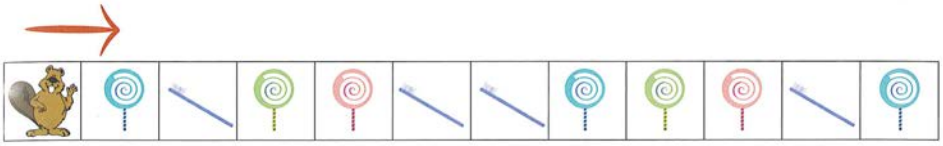
When two adjacent dogs switch positions, it is called a swap.
We use multiple swaps to arrange the three big dogs to be in three consecutive positions.
Question:
What is the minimum number of swaps that can be done to achieve this?

(A) 5 swaps (B) 6 swaps (C) 7 swaps (D) 8 swaps

3. Lollipop and toothbrush

The little beaver goes to a candy tunnel with a lollipop or a toothbrush at every other step. This passage can only go forward and not backward. Thus, at every step, the little beaver must decide whether to eat candy, brush his teeth, or keep going without doing anything, according to the things placed in that position. The little beaver cannot take the lollipop or toothbrush with him.

The little beaver wants to eat as much candy as possible but also wants to abide by dental health rules. Therefore, after eating a maximum of two lollipops, he must brush his teeth before eating the next lollipop.



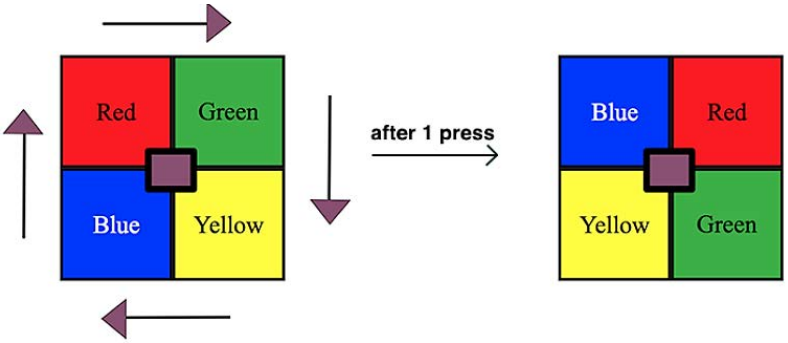
Question:

What is the maximum number of lollipops that the little beaver can eat in order to follow the dental health rules?

(A) 3 lollipops (B) 5 lollipops (C) 6 lollipops (D) 7 lollipops

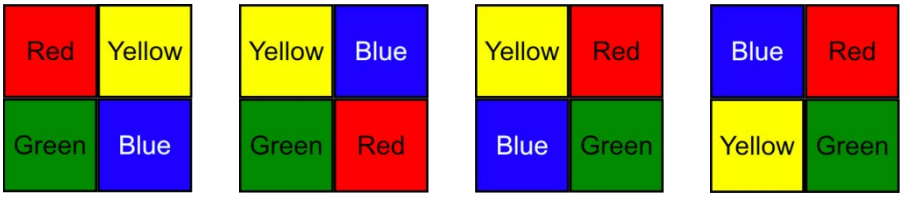
4. Spinning squares

Here is a simple push-square game. The colors rotate each time the center button is pressed.



Question:

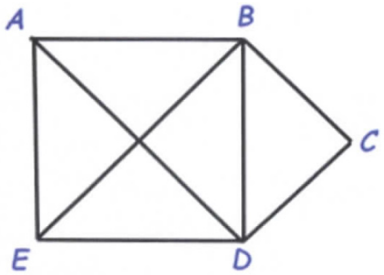
If we press the button one more time, where will the red, blue, green, and yellow squares be?



(A) (B) (C) (D)

5. Painting robot

The little beaver's tennis court looks like this odd shape when viewed from above.



The little beaver wants to use a robot to paint the tennis court. He hopes to paint each line in the court but, to save paint, not repeat any line. Additionally, the robots can only be turned off when the pitch is fully painted.

Question:

Which of the following paths should the robot take to paint each line without repeating any line?

(A) A→B→C→D→E→B→D→A→B

(B) A→D→B→E→D→C→B→A→E

(C) A→D→B→E→D→C→B→A

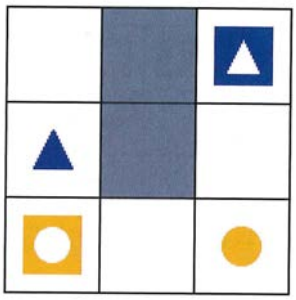
(D) E→A→D→E→B→C→D→E

6. Board game piece returns home

A computer game is played on a board comprising white and gray squares. Pieces can be placed on the white squares, but they cannot be placed on or moved into the gray squares.

The following two types of pieces are initially placed on the white squares:
circle, triangle, or rhombus.

The home of each game piece has a square with a hollowed-out shape in the center.

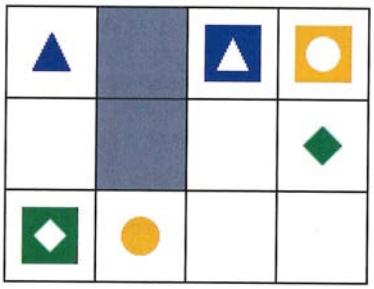


Each piece can be moved to adjacent squares using four commands: up, down, left, and right. The adjacent squares must be white, and no other pieces or other pieces's home. If a piece moves to its home, it and its home disappear from the board.

The goal is to return each piece to its home (leaving a board without pieces).
Taking the chessboard in the above picture as an example, the correct order for the shaped piece to move home is: circle: (left, left) and triangle: (bottom, right, right, top, top).

Question:

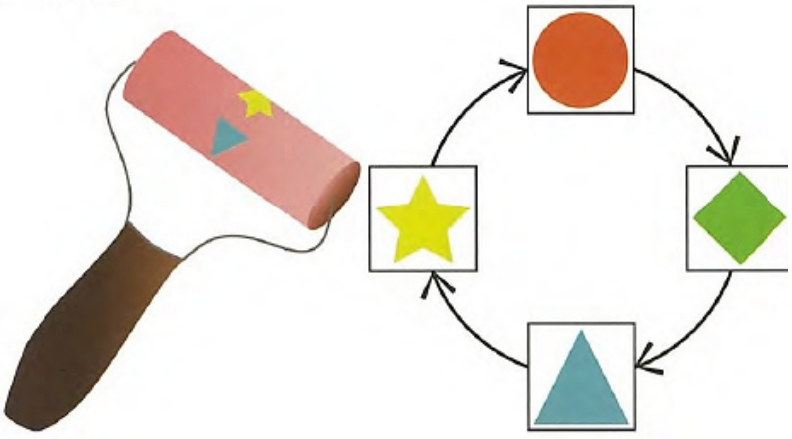
Which string of the following movement sequences will complete the task shown on the chessboard on the bottom?



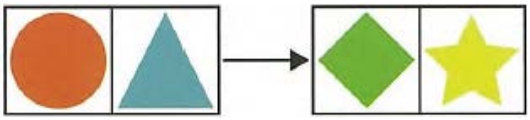
(A) Circle: (right, right) Rhombus: (left, bottom, right, right) Circle: (up, up) Triangle: (down, down, right, right, up, up)
(B) Rhombus: (left, down, left, left) Circle: (right, right, up, up) Triangle: (down, down, right, right, up, up)
(C) Circle: (right, right) Rhombus: (left, left, down, left) Circle: (up, up) Triangle: (down, down, right, right, up, up)
(D) Circle: (right, right) Rhombus: (left, down, left, left) Circle: (up, up) Triangle: (down, down, right, right, up, up)

7. Bebras painting

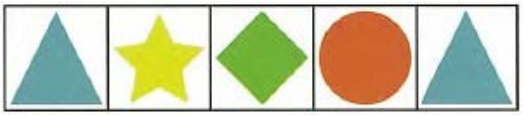
The beaver children have found a magic roller.
The roller replaces a shape in a painting with the next shape shown by the arrows below.



Example:
When Ben uses the magic roller to paint over the painting on the left, he gets the painting on the right.



Question:
What will the painting below look like after using the magic roller?



(A)

(C)

(B)

(D)

8. Dancer

Verity makes an animation of a man dancing. So far, she has only completed the first and last frames.
The man can only move one of his arms or legs at a time, and there should only be one difference between adjacent frames.
Question: Drag and drop the images into the correct empty frames to complete the animation.

(A)

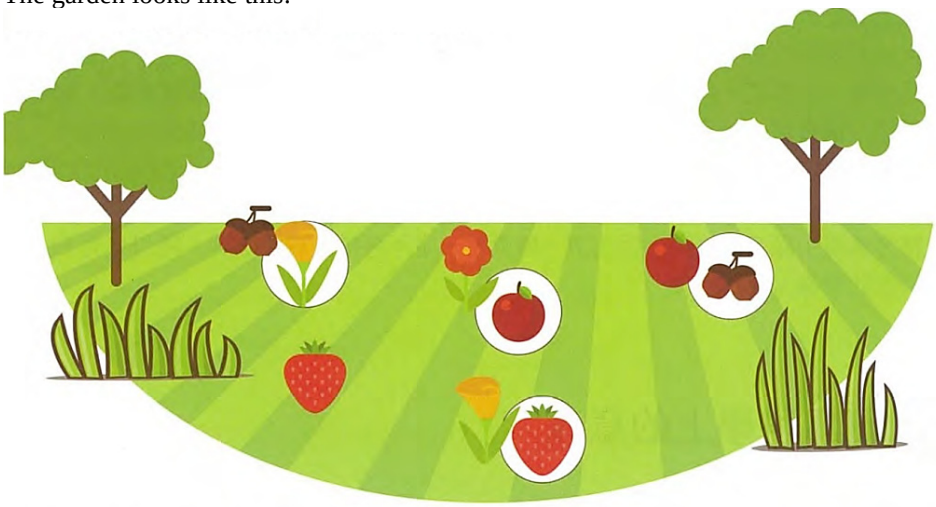
(C)

(B)

(D)

9. Secret recipe

Esther has asked Ivan to cook a special cake made of five ingredients.
She has placed labels next to the ingredients in the garden. However, one ingredient has no label.
The labels tell Ivan which ingredient must be added next in the sequence.
The garden looks like this:



Question:
Which ingredient should be added first?

(A)

(B)

(C)

(D)

10. Bakery

Beaver village's bakery bakes 3 bagels every 2 min.
There is a queue in front of the bakery, which only serves only 1 customer at a time. The bakery opens at seven in the morning and begins baking when the first customer orders. There were already 3 beavers queuing up when the doors opened.

The first beaver, Ali, needs to buy 7 bagels, the second beaver, Bilgin, needs 3 bagels, and the third beaver, Yasemin, needs 5 bagels.
However, beavers in the line can only buy a maximum of 3 bagels at a time. Thus, if they want to buy more, they must immediately go to the back of the line and line up again.

The third beaver, Yasemin, bought the 5 bagels he needs a few minutes after the business starts?

(A) 6 min (B) 8 min (C) 10 min (D) 12 min

Appendix II
Programming Empowerment Scale

Gender：□Male □Female
Grade：□4 □5
Have you been exposed to courses related to procedural design before?
□Yes □No
Which programming courses have you taken before?
□Scratch □Code.org □Alice □Java □Python □Others:_____

Items	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Meaningfulness					
Meaningfulness 1. Programming is useful to me.	☐	☐	☐	☐	☐
Meaningfulness 2. Programming will help me achieve my goals.	☐	☐	☐	☐	☐
Meaningfulness 3. I want to become good at programming.	☐	☐	☐	☐	☐
Meaningfulness 4. Programming is important to me.	☐	☐	☐	☐	☐
Impact					
Impact 1. I want to use programming to help solve problems in the world.	☐	☐	☐	☐	☐
Impact 2. I want to use programming to improve people's lives.	☐	☐	☐	☐	☐
Impact 3. I can use programming to make daily life easier.	☐	☐	☐	☐	☐
Creative self-efficacy					
Creative self-efficacy 1. I would like to design things using programming.	☐	☐	☐	☐	☐
Creative self-efficacy 2. Computer programmers are creative.	☐	☐	☐	☐	☐
Creative self-efficacy 3. It is important to be creative when you are programming.	☐	☐	☐	☐	☐
Programming self-efficacy					
Programming self-efficacy 1. I can learn how to program.	☐	☐	☐	☐	☐
Programming self-efficacy 2. I am good at programming.	☐	☐	☐	☐	☐
Programming self-efficacy 3. I think of myself as someone who can program.	☐	☐	☐	☐	☐
Programming self-efficacy 4. I have the skills to program.	☐	☐	☐	☐	☐
Programming self-efficacy 5. I have confidence in my ability to program.	☐	☐	☐	☐	☐
Interest in programming					
Interest 1. Programming is interesting.	☐	☐	☐	☐	☐
Interest 2. I am curious about the content of programming.	☐	☐	☐	☐	☐
Interest 3. I think the content of programming is fun.	☐	☐	☐	☐	☐
Interest 4. I am very interested in computer programming activities.	☐	☐	☐	☐	☐
Attitude toward collaboration in programming					
collaboration 1. I like to program with others.	☐	☐	☐	☐	☐
collaboration 2. I finish things faster when I program with others.	☐	☐	☐	☐	☐
collaboration 3. I have better ideas when I program with others.	☐	☐	☐	☐	☐
collaboration 4. People ask me for help with computers a lot.	☐	☐	☐	☐	☐