International Journal of Learning, Teaching and Educational Research
Vol. 20, No. 6, pp. 88-108, June 2021
https://doi.org/10.26803/ijlter.20.6.5

Improving Novice Students' Computational Thinking Skills by Problem-Solving and Metacognitive Techniques

Nor Hasbiah Ubaidullah, Zulkifley Mohamed, Jamilah Hamid and Suliana Sulaiman
Universiti Pendidikan Sultan Idris, Perak, Malaysia
https://orcid.org/0000-0002-4753-5382
https://orcid.org/0000-0003-1170-046X
https://orcid.org/0000-0003-2761-1074
https://orcid.org/0000-0002-2440-8831

Rahmah Lob Yussof
Universiti Teknologi MARA, Pahang, Malaysia
https://orcid.org/0000-0002-0650-9428

Abstract. Admittedly, the teaching and learning of programming courses in the computer science and information technology programs have been extremely challenging. Currently, most instructors depend on either the problem-solving technique or the metacognitive technique to help students develop a range of cognitive skills, including metacognitive skills, which are important in the development of a strong computational thinking skill required for 21st-century learning. Studies focusing on the practices of instructors in using both techniques are scarce, thus motivating the researchers to carry out this study. This study was based on a qualitative approach involving a case-study design in which five (5) male and five (5) female instructors were selected from 10 pre-university centers in Malaysia as the respondents and participants in an intervention program. The research instruments used were an interview checklist and intervention guidelines. As anticipated, the findings showed that the activities of each technique could only help students develop certain sub-skills of the computational thinking skill, thus underscoring the need for instructors to integrate both techniques in their teaching practices. Thus, it could be reasoned that using either the metacognitive technique or the problem-solving technique alone would not be sufficient to help students develop strong computational thinking skills, as each technique has its strengths and weaknesses. Therefore, it becomes imperative for instructors to leverage the strengths of both techniques by integrating both of them in the teaching and learning of programming courses.

Keywords: computational thinking skill; teaching and learning techniques; learning computer programming; programming teaching

1. Introduction

Lately, a growing number of researchers and scholars have highlighted the importance of computational thinking (CT) in programming, made evident by an increasing number of studies that focus on such a construct, such as studies by Margarida (2017) and Xabier et al. (2018). To date, CT skills have been widely researched in many developed nations; In Malaysia, however, studies of such nature have been scarce. Nonetheless, in recent years, many stakeholders in the educational sector have put greater emphasis on the importance of CT in education, particularly in computer programming. For example, the Ministry of Education of Malaysia integrated elements of CT in the school curriculum in 2017 (Ung, 2017). With this integrated curriculum, students would be able to learn basic computer science through activities involving problem-solving and logical thinking (Ministry of Education Malaysia, 2016). Specifically, the elements of CT, such as logical reasoning, estimation, developing the algorithm, and abstraction, were embedded in the new curriculum encompassing all levels of education from primary to tertiary education (Ministry of Education Malaysia, 2016), signifying that all the stakeholders were strongly aware of the importance of students to acquire this important skill. Lately, many Malaysian school teachers have gone through a series of workshops to train them methods that they could use to teach students such a concept. The same is, however, not forthcoming for college and university lecturers; thus, it is hardly surprising to see them having difficulties in identifying effective methods that they could use to help develop strong CT skills among their students. As such, appropriate measures are needed to mitigate this predicament to ensure they could effectively teach students to become digital makers in Malaysia (Aizyl, 2016; Joseph, 2016).

As such, it is vital to carry out more studies on the development of CT by focusing on appropriate activities in the learning of information technology and computer science, particularly programming courses. Surely, the findings of such studies could inform practitioners of the appropriate teaching techniques that they could use to help students develop such skills (Filiz, 2016). Admittedly, many studies have been carried out thus far, but they were primarily focused on metacognitive and problem-solving skills needed for programming (Havenga, 2015), with only a handful being dedicated to investigating their relations to CT. To date, a growing number of local researchers have carried out several studies that dealt with such a research focus (Mohd Rum, 2015; Ung, 2017). Such studies, however, mainly centered on the preparation of schools' teachers who would implement teaching activities that could enhance students' CT. By contrast, there is a dearth of similar studies that focus on the preparation of college and university lecturers that would them choose proper teaching and learning techniques to help improve undergraduates' and college students' CT.
Hence, this research was conducted to address such a research gap by focusing on metacognitive, problem-solving, CT skills, and programming. This study was premised on the teaching and learning of information technology in general and programming particularly among novice learners by examining appropriate techniques with which lecturers could use to enhance students' CT skills. In this study, metacognitive and problem-solving skills were integrated into the

teaching and learning of a programming course. Thus, to direct research, two research questions have been formulated:

1. How would lecturers apply metacognitive and problem-solving techniques in the computer programming teaching environment to improve novice students' CT skills?
2. How would metacognitive and problem-solving techniques be embedded in the learning of a programming course to improve novice students' CT skills?

This study is crucial as it is an initial step for instructors in developing CT skills among students. As emphatically noted by Malaysia's premier, students need to have well-developed CT skills that would help nations in developing a capable generation of digital makers (Aizyl, 2016; Joseph, 2016).

2. Literature Review

Nowadays, more and more people need to depend on computational thinking (CT) skills to perform a broad spectrum of tasks. This skill is indispensable given its importance in today's information-driven societies, entailing people to think logically, analytically, and systematically in solving numerous problems (Swaid, 2015). According to Park (2016), CT involves problem-solving skills in programming with the creative use of computer hardware and software. Specifically, according to Wing (2016), individuals tap on their CT as they try to solve a complex problem by mathematically decomposing it into small units. From the learning perspective, the same researcher argues that students with good CT would be able to create command-line algorithms for a computer to perform a specific task in solving a particular problem. In other words, they reason that CT skills are closely related to problem-solving skills by using the computer. More broadly, it can reasoned that CT skills are associated with problem-solving skills without the use of the computer in various fields, such as science and mathematics. For example, CT skills can also be developed through the teaching of sciences or languages at the elementary school level. As such, such skills should be viewed in a more diverse, multi-disciplinary context.

According to Mannila et al. (2014), CT is a term that encompasses a set of concepts and processes of computer science involved in deriving solutions to problems in several disciplines. In essence, according to Selby (2015), CT is defined as a multi-faceted skill comprising several sub-skills, namely abstraction, decomposition, evaluation, generalization, and algorithmic thinking. As such, the concepts of CT can be implemented in the classroom, such as logic, algorithm, decomposition, pattern, abstraction, and evaluation (Barefootcas, 2014). The definitions of such a concept made by Mannila et al. (2014), Selby (2015), and Barefootcas (2014) underscore that CT, which is an extremely important skill in the computing field, can also be applied to other important fields of knowledge.

Over recent years, several scholars, including Margarida et al. (2017), Denning (2017), Buitrago Flórez et al. (2017), and Xabier et al. (2018), have embedded such concepts in the computer science and information technology fields through programming courses, given their close relationships. As emphasized by

Margarida et al. (2017), CT skills can be developed and assessed through ill-defined problems at different educational levels. Essentially, CT skill concerns one's ability to analyze problems, make informed decisions, and solve problems creatively (Kafai, 2016). As such, from a practical standpoint, students' abilities in solving problems through logical thinking and in writing codes for computer applications and hardware are a measure of their CT (Djambong & Freiman, 2016). In this respect, logical thinking and the skill in writing codes are two important elements in learning programming. Given that the elements of CT (namely abstraction, decomposition, evaluation, generalization, and algorithmic thinking) are closely related to the skills needed in solving computer programming, such as logical thinking, CT can, therefore, be developed through the teaching of computer programming.

Moreover, programming for applications of systems and hardware in diverse fields can serve as a medium of CT (Voogt et al., 2015). More importantly, programming is a critical discipline of knowledge that students of computer science and information technology must master. However, most students find programming to be a difficult subject or course to learn, let alone to master it (Nurul Faeizah et al., 2020). Specifically, students face difficulties in solving programming problems that entail them to have the proper skills to deal with the problem-solving, syntax, and semantic of a programming language (Malik & Cildwell-Neilson, 2017; Hooshyar et al., 2015). As observed by many researchers, most students, in particular novice students, lack critical thinking and problem-solving skills to help them learn computer programming (Djambong & Freiman, 2016; Poli & Koza, 2014). In view of the challenges facing students in learning computer programming, teachers and instructors need to be creative in their teaching to help stimulate students to think logically and critically, which will certainly help them solve computing problem effectively and efficiently.

Essentially, critical thinking refers to the use of cognitive skills or strategies to achieve an intended outcome. In other words, critical thinking is purposeful, reasoned, and goal-directed mental process for solving problems, formulating inferences, calculating likelihoods, and making decisions (Halpern, 1999). As emphasized by Ramdiah and Duran (2014), critical thinking involves the skillful handling of the structure inherent in thinking by imposing intellectual standards upon it. On the other hand, Paul et al. (1993) argued that critical thinking is self-directed, self-disciplined, self-monitored, and self-corrective thinking. The same scholars assert that the capacity to solve problems is an important aspect of analytical thought, which encompasses general problem-solving interventions, including problem recognition, problem description, approach planning, the organizing of knowledge and resource distribution, tracking, and assessment (Sterberg & Sterberg, 2012). In the context of learning programming, the problem-solving steps required include the identification and definition of problems, the planning of problem-solving, the design of problem-solving, coding, testing, and documentation. In this regard, problem-solving skills refer to the ability to solve problems accurately, identify and define problems, propose alternative solutions, test and select the best alternative, and implement the selected solution. On the other hand, critical thinking relates to self-directed, self-disciplined, self-monitored, and self-corrective thinking. Juxtaposing the above two skills, it can be clearly seen that critical thinking skills and problem-solving skills are closely related to one another.

Certainly, a lack of problem-solving skills contributes to students' poor performances in learning programming (Mohd Rum, 2015), a subject matter consisting of complex, abstract concepts that make it difficult for students to understand, interpret, and perform complex tasks (Malik & Coldwell-Neilson, 2017). Thus, they must have strong analytical, logical, and problem-solving skills as well as the skill in learning a particular programming language. The latter is important because programming allows students to analyze their thought processes and their strategies as a cognitive exercise that encourages the method of applying a newly learned solution to new problems (Mohd Rum, 2015). The practice of problem-solving in computer programming can improve students' cognitive skills, allowing them to work methodically to build representations (Mayer, 2003). Such a practice provides the opportunity to help students develop strong metacognition. For example, Bergin et al. (2005) signified that students who achieved well in programming were more reliant than low-performing students on metacognitive management techniques, thus underlining the value of having strong metacognitive abilities for students to help them learn to program.

Metacognition refers in essence to the deliberate preparation, monitoring, and assessment of the cognitive processes of individuals, such as their emotions, as they participate in the learning process (Sterberg & Sterberg, 2012). On the one hand, metacognitive knowledge refers to a more advanced level of knowledge that allows students to monitor, handle, interpret, and understand their knowledge during the learning process (Gaeta, 2014; Nimmi & Zakkariya, 2016). According to Abdullah et al. (2017), metacognition consists of two parts, namely the knowledge component and the skill component. Between these two components, the latter is deemed more important in the learning of computer programming as it helps students to effectively engage in problem-solving activities in which they learn to solve programming problems. Thus, metacognitive strategies or skills are critical to effective learning as they influence the control of cognition in activities involving planning, orienting, monitoring, checking, selecting, revising, evaluating, self-monitoring, and self-evaluating. As emphasized by Brown (1992), students need to have strong metacognition to enable them monitor their processes of thought strategically and effectively. Therefore, teachers' instructions and feedback in the teaching and learning process would have a profound impact on the development of students' metacognitive skills (Veenman, 2006; Hinojosa, Rodriguez & Paez, 2020).

Despite the plethora of studies on the use of problem-solving and metacognition techniques in computer programming, not many studies have been devoted to studying their impacts on the development of CT skills. Thus far, only a handful of such studies has been carried out in Malaysia, which mainly involve the applications of such techniques at the school level that focused on teachers' readiness in implementing the techniques. Therefore, this study was carried out that focused on the teaching techniques that could be used to improve pre-university students' CT skills. Specifically, this study aimed to examine the impacts of the integration of problem-solving and metacognition techniques on the improvement of CT skills among novice programming students.

3. Methods

This qualitative study was based on a case study research design involving a series of semi-structured interviews in which several computer programming instructors were interviewed. In the interviews, information regarding their teaching activities before and after a learning intervention based on specific metacognitive and problem-solving guidelines was gathered to address the research questions.

3.1 Procedure

The procedure of the data collection of this study was slightly adapted from that used by Havenga (2015) who used a series of interviews that were carried out before and after an intervention. Such a slight adaptation was made to suit the context of this study. In particular, this study consisted of two phases, namely Phase 1 and Phase 2. The former involved collecting data and information on CT, while the latter concerned eliciting lecturers' opinions regarding their teaching practices before and after the intervention, which helped highlight any teaching changes that might have occurred.

The following are the two phases of the activities carried out in this study.
Phase 1*:* In the first phase, a critical review of the current literature was performed to help determine aspects of CT that need further research. The review of the literature was primarily centered on relevant studies published in books, research papers, conference proceedings, and journal articles.

Phase 2*:* In the second phase, a case study was carried out where the researchers interviewed several instructors to elicit their feedback on their teaching practices before and after a learning intervention, which focused on helping to enhance novice programming students' CT skills. This approach enabled the researchers to determine if there was a substantial change in their teaching practice, which is in line with recommendations made by Gill (2011).

3.2 Respondents

The respondents of the study were made up of ten (10) instructors of a programming course taught at several pre-university colleges, who were recruited through the purposeful sampling technique. They were primarily selected due to their involvement in the teaching of programming in which metacognitive and problem-solving techniques were used. Each one holds a Master Science's degree in either computer science or information technology and had a teaching experience of at least five (5) years. Table 1 summarizes the demographics of the selected instructors.

Table 1: The Demographics of the participants

Participant ID	Gender	Age (in a year)	Working experience (in a year)
P1	Female	39	9
P2	Male	44	15
P3	Female	40	7
P4	Male	40	17
P5	Male	45	8
P6	Female	38	5
P7	Male	42	12
P8	Male	40	15
P9	Female	50	25
P10	Female	48	23

3.3 Data collection

Data were gathered through a series of semi-structured interviews involving programming instructors, which were conducted two times to elicit appropriate information on their teaching practices before and after the learning intervention. The following sub-sections provide a detailed account of the pre-intervention interviews, learning intervention, and post-intervention interviews.

1. **Pre-intervention interviews**: The pre-intervention interview sessions were carried out one week before the intervention program to help determine whether the programming instructors had taught their students any metacognitive and problem-solving skills.

2. **Learning intervention***:* In the intervention, the instructors were guided to perform specific steps to derive proper solutions to programming problems based on problem-solving guidelines and metacognitive skills as follows:

- Carefully read a given problem, highlight the main ideas, and comprehend and write down the main requirements of the problem. Review and refine such ideas and requirements as needed.
- Formulate a solution to the problem.
- Spell out the details of the required steps in terms of appropriate inputs, processing, and outputs. Highlight their aims and the processes involved.
- Go through the solution that you have proposed.
- With a given programming language, code all the above elements into a program. Examine your program for any programming errors and carefully evaluate the steps that you have performed.
- Test your program.
- Carefully review the programming codes and programming semantics.
- Determine how effective your solution and explain whether it is the best solution.

3. **Post-intervention interviews**: The final interviews were carried out to elicit information regarding instructors' overall experiences in the intervention program. They were prompted with the following question: In what way would your experiences in using metacognitive and problem-solving skills relate to your teaching approach that could help improve your students' programming and CT skills?

4. Results

The findings of the study are discussed based on two themes, namely Theme 1 (Using problem-solving competence in programming teaching and learning) and Theme 2 (Using metacognitive competence in programming teaching and learning). In turn, the discussions of the two themes are divided into two sub-sections, namely before the intervention and after the intervention.

4.1 Theme 1: Using Problem-Solving Competence in Programming Teaching and Learning

1. **Before Intervention***:* The feedback of the instructors indicated that they used specific problem-solving activities in their teaching practices before the intervention. For example, the first, second, seventh, and tenth participants, P1, P2, P7, and P10, stressed the analysis of questions by instructing their students to determine the appropriate input, process, and output during the planning of their programs, as exemplified by the second instructor's feedback regarding his students' work as follows: "*… the students first construct the input, process, and output (IPO) table, and then they jot down the required steps*. Also, they emphasized the use of algorithms as part of the detailed planning of programs, as highlighted by the same participant who gave the following feedback: *"algorithms are essential to solving programming problems effectively. Hence, after constructing the IPO table, they should perform the algorithms before moving to the next steps"*. Even though the third, fifth, and eighth participants, P3, P5, and P8, did submit their students' homework, there was no mention of any specific problem-solving activities used by their students in solving the programming problem. By contrast, the sixth and ninth participants, P6 and P9, confessed that they did help their students by demonstrating the proper steps in analyzing programming problems.

2. **After Intervention**: Once they had undergone the intervention, the instructors put a greater emphasis on the detailed requirements for each problem-solving step. For example, P3's comments were highly informative as follows: "*After they had split the problem into several sub-problems, the students could formulate a solution, as it became easier for them to manage the sub-problems as opposed to dealing with the main problem*." Moreover, the use of the guidelines proved to be extremely helpful, made clear by the same participant's comment as follows: "*The students were compelled to think critically and logically as they tried to solve the problem, entailing them to perform the appropriate steps in developing a program*". Likewise, the fourth participant's (P4) remark was equally compelling when he said the following words:

"*Students must familiarize the first step before attempting to perform the ensuing steps that lead to the final solution*". Also, he elaborated on some strategies that could be used to deal with programming problems. By contrast, the fifth participant (P5) stressed the importance of time that students should take into consideration in analyzing problems, as clearly highlighted by his comment as follows: "*Spending more time in analyzing the problems helped students to gain a better understanding, which led to better solutions*". Table 2 summarizes the problem-solving activities deemed highly effective by the instructors in the teaching and learning of programming before and after the intervention.

Table 2: Highly emphasized problem-solving activities in the teaching and learning of programming before and after the intervention

Before Intervention		After Intervention	
Participant	**Activities**	**Participant**	**Activities**
Participant 1	Planning - analysis (IPO) Design - algorithm	Participant 1	Planning - analysis (IPO) Design - algorithm
Participant 2	Planning - analysis (IPO) Design - algorithm	Participant 2	Planning - analysis (IPO) Design - algorithm
Participant 3	Assigning homework - (specific activities were not mentioned)	Participant 3	Planning - breakdown the problem - think critically and logically
Participant 4	- not available	Participant 4	Observation Discussion - strategy on how to approach a programming problem
Participant 5	Assigning homework - (specific activities were not mentioned)	Participant 5	Analysis - need time when performing analysis
Participant 6	Analysis - discuss the analysis of programming problems	Participant 6	Analysis - discuss the analysis of programming problems
Participant 7	Planning - analysis (IPO) Design - algorithm	Participant 7	Planning - analysis (IPO) Design - algorithm
Participant 8	Assigning homework - (specific activities were not mentioned)	Participant 8	Planning - breakdown the problem - think critically and logically
Participant 9	Analysis - discuss the analysis of programming problems	Participant 9	Analysis - discuss the analysis of programming problems
Participant 10	Planning - analysis (IPO) Design - algorithm	Participant 10	Planning - analysis (IPO) Design - algorithm

As clearly shown, a majority of the instructors used problem-solving activities, namely analysis, planning and design, in their teaching of programming before and after intervention. For example, only Participant 6 and Participant 9 used the analysis activity in their teaching before intervention. Also, Participant 1, Participant 2, Participant 7, and Participant 10 used the planning and design activities in their teaching of programming. However, after intervention, more participants used all the problem-solving activities in their teaching of programming. In particular, the number of participants who used the analysis activity increased to three (3). This was made evident by Participant 5, Participant 6 and Participant 9, who previously had never used such an activity, had now used the analysis activity in their teaching. Likewise, the number of participants who used the planning and design activities had increased to six (6), as exemplified by Participant 1, Participant 2, Participant 3, Participant 7, Participant 8, and Participant 10 who used such activities in their teaching after the intervention.

Admittedly, performing these activities entail students to have good programming skills. Furthermore, these activities are closely related to the elements of CT skills. For example, the analysis, design, and planning of problem-solving activities are closely related to the abstraction, algorithm, and decomposition and generalization of CT, respectively. As such, performing the former activities can help students enhance the latter skills. Given these revelations, it is, therefore, important for programming lecturers to embed such problem-solving activities in the teaching of computer programming, the impacts of which can enhance both students' programming skills and CT skills.

4.2 Theme 2: Using Metacognitive Competence in Programming Teaching and Learning

1. **Before Intervention***:* The feedback elicited showed that the respondents also relied on some form of metacognitive skills in their teaching practices before the intervention. For example, the first participant (P1) allowed her students to plan their solutions before writing codes, as evidenced by her comments as follows: *"Usually, I would discuss the problem first with my students by asking them to analyze the algorithms before writing the essential codes. Hence, they wrote the codes on a piece of paper before coding those algorithms on the computer"*. On the other hand, the fourth participant (P4) allowed his students to explore the detail of a new topic based on the belief that the students could direct their thinking processes, made clear by his comment as follows: *"I made a point to always to encourage my students to use their creativity in coding"*. By contrast, the sixth participant (P6) relied on other strategies, namely problem-based learning, and collaborative learning, to help guide her students' self-directed learning activities.

2. **After Intervention:** As prescribed by the guidelines of the intervention program, the instructors gave some examples of the metacognitive skills that they had taught in the classroom. In particular, the first, second, and fifth participants (P1, P2, and P5) stressed the importance of planning a solution before writing a program, which could be discerned by some of their comments, such as those made by the first participant (P1) as follows:
"The greater their efforts in planning, the greater they could understand the question … the students could tackle the question quite easily. I think most of them were able to do just that" and *"… therefore, to each question, every student was prompted to ask, 'What must I have to do to answer this question?'"*. Likewise, the second participant's (P2's) comments were also revealing based on the statements he made as follows:

"*Students should do the planning before they wrote the program, as it was very important … they simply could not go to the computer and write the coding*". The above comments made by the first and second supported students' mental activities by guiding them to refocus on a problem in hand, while the fourth participant (P4) emphasized the importance of such activities based on his feedback as follows: "*It certainly helped them to fully grasp the problem*". The third participant's (P3's) stressed the importance of scaffolding as highlighted by her comment as follows: "*At first, I guided my students. Then, as they could understand the problem and had some ideas on how to solve it, I let them continue with their work*". Interestingly, the sixth participant (P6) asserted that he had to divide students into several groups to facilitate them to discuss their problems more effectively. As a whole, the above findings are consistent with Francom (2010) and Mohd Rum (2015), who found subject-matter knowledge and self-directed learning skills (e. g. metacognition) collectively helped students to manage their thinking processes. Table 3 summarizes the metacognitive activities that the participants deemed important in the teaching and learning of programming before and after the intervention.

Table 3: Highly emphasized metacognitive activities in the teaching and learning of programming before and after the intervention

Before Intervention		**After Intervention**	
Participant	**Activities**	**Participant**	**Activities**
Participant 1	Planning - provided a chance for students to plan their solutions	Participant 1	Planning - students involved actively in their programming tasks
Participant 2	- not available	Participant 2	Planning - students involved actively in their programming tasks
Participant 3	- not available	Participant 3	- scaffolding
Participant 4	Discovering - students discovered details. -encouraged students to be creative	Participant 4	Planning - students involved actively in their programming tasks
Participant 5	- not available	Participant 5	Planning - students involved actively in their programming tasks
Participant 6	Using additional strategies - problem-based learning - collaborative learning (enhanced self-directed learning)	Participant 6	Discussion - supported students by group Discussion

Participant 7	Planning - provided a chance for students to plan their solutions	Participant 7	Planning - students involved actively in their programming tasks
Participant 8	- not available	Participant 8	Planning - students involved actively in their programming tasks
Participant 9	- not available	Participant 9	- scaffolding
Participant 10	Discovering - students discovered details. -encouraged students to be creative	Participant 10	Planning - students involved actively in their programming tasks

As shown in Table 3, the only metacognitive activity used by the instructors was planning, as indicated by Participant 1 and Participant 7 who practiced such an activity in their teaching before the intervention. The remaining participants, however, used other types of teaching activities that were related to those of problem based-learning and collaborative learning. Interestingly, after the intervention, more participants used the metacognitive activities in their teaching of programming. Specifically, the number of participants who used the planning activity increased to seven (7), as demonstrated by Participant 1, Participant 2, Participant 4, Participant 5, Participant 7, Participant 8, and Participant 10. Arguably, the planning activity is one of the important steps in programming that every programming student need to learn and master. Moreover, such an activity can also help enhance students' decomposition and generalization abilities, which constitute two of the components of CT skills. Therefore, by performing the planning activity, students will be able to enhance both their programming skills and CT skills.

5. Discussion

The analysis of respondents' feedback based on the first theme (Theme 1) showed that only three instructors had integrated problem-solving activities, namely analysis, and planning, in their teaching practices before they followed the intervention program. As anticipated, after the intervention, more instructors indicated that they had integrated more specific problem-solving activities in their teaching, which could be attributed to their compliance with the guidelines given to them. Arguably, most instructors not only knew but also were quite conversant with a problem-solving technique that is widely regarded as the most popular technique in teaching and learning programming, thus compelling them to integrate it into their teaching practices. Such findings are consistent with Malik and Coldwell-Neilson (2017), Hooshyar et al. (2015), and Mohd Rum (2015), most of whom assert that problem-solving is an effective strategy to help students to understand and solve programming problems. Also, the same findings showed that majority of the instructors utilized analysis and planning activities in their teaching after the intervention. Oddly, only one instructor used observation and discussion activities in teaching his students.

In this regard, the discussion technique could serve as an effective means to help students, especially novice students, learn to program as they could suggest creative ideas and work collaboratively to solve programming problems. Certainly, such a learning process could help them improve their logical thinking. As such, instructors had to be resourceful and creative in stimulating effective discussions among their students. As revealed in this study, a combination of problem-solving and discussion techniques could help train novice students to enhance their logical thinking and problem-solving skills. Such findings are consistent with those of previous studies, signifying such techniques as highly effective (Malik & Coldwell-Neilson, 2017; Uysal, 2014).

Essentially, solving a given programming problem entails students performing several activities, namely analysis, planning, design, coding, and evaluation, which were given strong emphasis by the instructors as shown in Table 2. Such emphasis was not unexpected as these activities are critical components of problem-solving techniques that students had to apply before carrying out other ensuing activities. In the analysis activity, students were required to correctly identify the input, process, and output, which were essential to helping them to enhance their abstraction skills by identifying and extracting information that could help define the main idea of a given problem. On the other hand, the planning activity could help improve their decomposition skills, enabling them to split problems into smaller, manageable parts. As such, the above two activities, namely, analysis and planning, are deemed important steps in problem-solving. Therefore, it was not surprising to note that most instructors paid strong attention to these two activities to ensure their students would be able to perform other ensuing activities, namely, design, coding, and evaluation. Put simply, by performing these two activities, students would be able to improve their abstraction (Seong-Won & Youngjun, 2020) and decomposition skills, which are two critical sub-skills of the CT skill (Román-González, 2017). Such findings are consistent with that of Mohd Rum (2015), indicating that teachers' instructions of management processes and activities can help students improve their learning performances.

The remaining activities, namely design, coding, and evaluation, are equally important in solving programming problems. In the design activity, students had to determine the proper steps to perform in the right sequence in solving the given problem. Surely, a high level of logical reasoning is required to solve problems by visualizing algorithms in a mental picture. In this regard, they use the algorithms and logical thinking concepts of CT. The next stage of programming is the implementation stage, which consists of several activities involving coding, compilation, linking, running, and debugging a program, necessitating strong logical thinking involving coding and identifying and correcting errors, which collectively could help improve students' CT in terms of algorithmic thinking skill.

The final stage of programming involves the evaluation activity, which is needed to test and validate a program. After being verified to be free of any error, the program needs to be tested with different inputs to ensure it could fulfill the requirements of a given problem and produce accurate output. In principle, this activity is equivalent to the concept of evaluation in CT. As asserted, novice students' CT skills could be enhanced through learning programming with the use of the above teaching techniques. Such an assertion is echoed by other scholars, such as Brennen and Resnick (2012), who argue that knowing about concepts and processes of computer programming could help students develop their CT skills or strategies. Furthermore, many researchers acknowledge that students could enhance their computational skills by engaging in certain activities, such as games, which require the use of some programming languages (Lee et al., 2014). Hence, instructors need to include such activities in their teaching strategies.

For the second theme (Theme 2), the findings showed that only three participants had integrated metacognitive activities in their teaching practices before following the intervention program. Revealingly, they indicated that they used problem-based learning and collaborative learning in their teaching to help improve students' self-directed learning, enabling the latter to perform self-monitoring and self-evaluation. By following the guidelines (that they had learned in the intervention program), all the instructors stated that they integrated relevant activities in their teaching practices, such as planning, scaffolding, discussion, and evaluation, all of which are similar to the activities of problem-solving technique. In particular, they emphasized the importance of scaffolding, which is a critical component to support students who are struggling in the early stage of learning (Feyzi-Behnagh et al., 2014).

Despite their claims of integrating metacognitive activities in their teaching, anecdotal evidence showed that they did not perform self-monitoring and self-evaluation activities to allow their students to reflect on the programming codes and semantics they had written. Most preferably, they should have prompted their students with some probing questions as follows: "How confident are you that you have effectively solved the problem?" or "Is this the best solution?". As a whole, the above findings helped the researchers to answer the first research question, namely "How would instructors and lecturers apply metacognitive and problem-solving skills in the teaching computer programming environment to improve students' CT skills".

The following discussions helped the researchers to answer the second research question, namely "How would the techniques of metacognitive and problem-solving be embedded in the learning of programming to improve students' CT skills?" and Table 4 shows the mapping of metacognitive and problem-solving activities with the components or elements of CT skill.

Table 4: The mapping of metacognitive and problem-solving activities to CT

Techniques/ activities	**Computational Thinking**				
	Abstraction	Decomposition	Generalization	Algorithm	Evaluation
METACOGNITIVE					
1. Planning		√	√		
2. Monitoring		√	√		
3. Selecting				√	
4. Checking				√	
5. Evaluating					√
6. Self-monitoring	Important for the development of students' thinking skill and social skill				
7. Self-evaluating					
PROBLEM-SOLVING					
1. Understanding and defining	√				
2. Planning		√	√		
3. Designing				√	
4. Coding		√		√	√
5. Testing			√		√

1. **Abstraction**: As indicated in Table 4, only the first element of the problem-solving technique, namely understanding and defining a problem, would significantly contribute to the development of the first element of critical thinking skill, namely abstraction. As such, students had to correctly identify and extract relevant information to enable them to define the main idea of a given problem. Such a process could certainly help enhance their abstraction skills (Shamir et al., 2019). This assertion parallels that of Soumela and Stavros (2014), who argue that abstraction is the method of making something straightforward from something complex by leaving out the unnecessary information, identifying the necessary patterns, and extracting concepts from concrete details.

2. **Decomposition:** As shown in Table 4, the planning and monitoring activities of the metacognitive technique and the planning activity of the problem-solving technique have a significant impact on the development of the decomposition skill, which is the second element of CT skills. In particular, students could perform the former activities by splitting a given problem into several manageable sub-problems, which closely mirrors those activities carried out by the decomposition process that breaks down a problem into smaller parts that are easier to deal with Shamir et al. (2019). Hence, by performing planning and monitoring activities of the metacognitive technique, students would be able to enhance their decomposition skills. Likewise, the coding activity of the problem-solving technique could wield a significant impact on the development of such a skill, as coding is an activity in which students write codes using a programming

language, which entails them to divide a major programming routine into smaller sub-routines.

3. **Generalization***:* As illustrated in Table 4, the planning and monitoring activities of the metacognitive technique would significantly contribute to the development of generalization skills of CT. Through such activities, students must rely on their prior knowledge in planning appropriate ways to solve a given problem and to adapt or reuse original codes to solve the problem. Similarly, the testing activities of the problem-solving technique could also contribute to the development of students' generalization skills, entailing them to run a program repetitively by using numbers of different inputs to derive an optimal output or a solution. In this respect, many researchers have emphasized the importance of this technique, such as Soumela and Stavros (2014) and Xabier et al. (2018). Therefore, students would be able to enhance their generalization skills by performing the above activity.

4. **Algorithm**: Table 4 shows that selecting and checking activities of the metacognitive technique and designing and coding activities of the problem-solving technique would have a profound impact on the development of students' algorithm skills of CT. In the selecting activity, students had to identify and select the most efficient and effective method of solving a given problem. In the checking and designing activities, they are required to carry out several activities as follows: (i) writing appropriate algorithms based on the outcomes of the analysis and planning activities, (ii) checking the algorithms that have been selected to ensure solutions generated therefrom would be effective, and (iii) checking the programs for syntax errors. As the algorithm skill of CT refers to the writing of step-by-step, precise, and explicit commands for the method (Buitrago Flórez et al., 2017; Nor Hasbiah & Jamilah, 2019), performing the above activities could effectively help students to develop this important skill in learning programming.

5. **Evaluation***:* Lastly, the evaluating activity of the metacognitive technique and the coding and testing activities of the problem-solving technique would have a significant influence on the development of students' evaluation skills, which are one of the important sub-skills of CT skill. According to Malaysia Digital Economy Corporation (2018), evaluation is the process of ensuring that a solution is good and suits a function, whether an algorithm, method, or process. Therefore, by carrying out such activities, such as evaluating programming outputs based on the readability and efficiency criteria, students could certainly develop their evaluation skill, which is the last component of computational skill, which is extremely important in helping students to develop efficient, effective programs. Given such importance, it becomes imperative for instructors to prioritize such activities in their teaching activities. As illustrated in Figure 1, some activities of metacognitive and problem-solving techniques do overlap with one another. Revealingly, none of the metacognitive activities has an impact on the development of the abstraction skill. Nonetheless, such a skill could be developed through the understanding and defining activities of the problem-solving technique. Likewise, the coding activity of problem-solving technique could help develop students' decomposition, algorithm, and evaluation skills of CT. Surprisingly, the self-monitoring and self-evaluating activities of the

metacognitive technique do not play a vital role in the development of any component of the CT skill. Nevertheless, these two activities are critical to helping students to develop strong self-management skills and social skills. As indicated, the problem-solving technique does have some activities that could help students develop those two skills. Admittedly, such overlapping gives rise to the need for the integration of metacognitive and problem-solving techniques in a way that they could effectively complement one another.

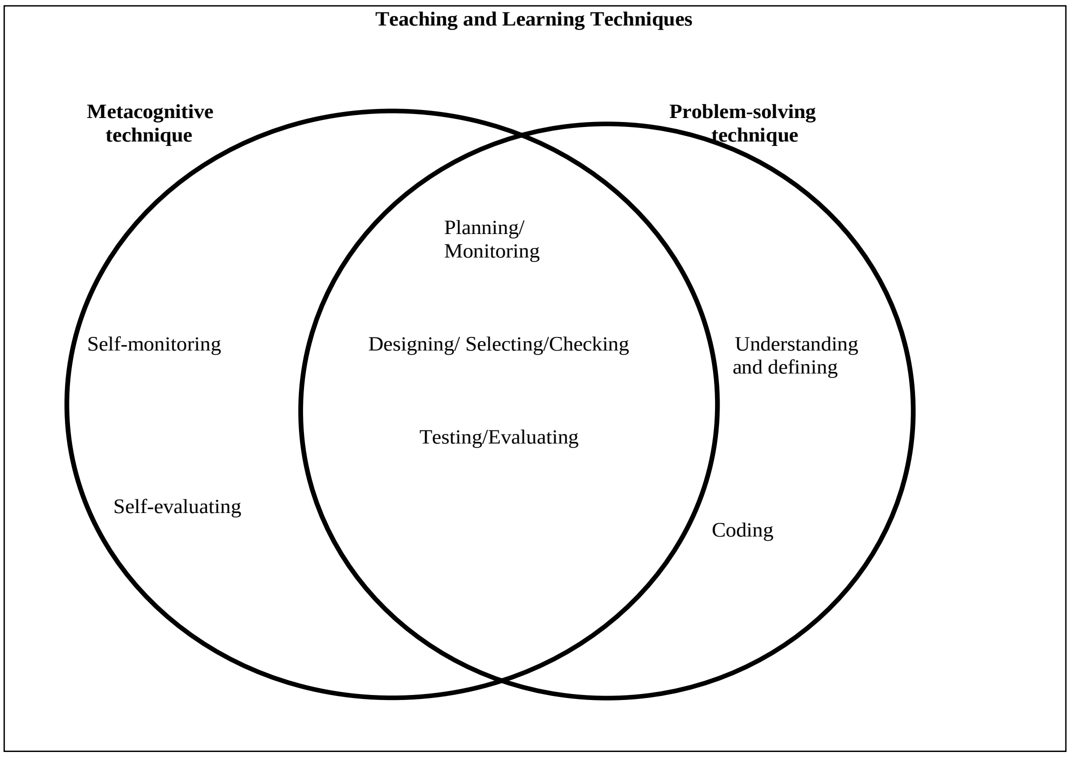
Teaching and Learning Techniques
Metacognitive technique
Problem-solving technique
Planning/ Monitoring
Self-monitoring
Designing/ Selecting/Checking
Understanding and defining
Testing/Evaluating
Self-evaluating
Coding

Figure 1: A Vann's diagram of activities of the metacognitive and problem-solving techniques

Additionally, instructors could perform self-monitoring and self-evaluating activities by prompting students to focus on their learning by making them ponder some apt questions, such as "*Have I made improvements in this area*?", "*What are my strengths*?", "*Are there rooms for improvement*?" and "*As the whole, where do I stand*?" With such questions, students could self-reflect and assess their understanding of the activities they had undertaken (Joseph et al., 2016; Nunaki et al., 2019). Arguably, in such activities, they could evaluate their levels of CT that they might have acquired (Buitrago Flórez et al., 2017; Filiz, 2016). Likewise, they could also learn about their weaknesses (if any) and take appropriate corrective measures. As discussed, it could be reasoned that using either the metacognitive technique or the problem-solving technique alone would not be sufficient to help students develop strong CT skills, as each technique has its strengths and weaknesses. Therefore, it becomes imperative for instructors to leverage the strengths of both techniques by integrating both of them in the teaching and learning of programming courses.

6. Conclusion and future work

As acknowledged by most researchers and scholars, CT skill is one of the competencies deemed critical in the learning environment of the 21st-century.

Thus, efforts are needed to develop and strengthen this vital skill among students. In this regard, computer programming courses could serve as a practical platform to help students acquire such an important skill. As demonstrated in this study, programming instructors could use problem-solving and metacognitive techniques to help students develop their CT skills. However, the former lacks self-monitoring and self-evaluating activities to improve students' self-management and social skills. By contrast, the latter lacks activities that could help students acquire strong abstraction skills. Given these drawbacks, both techniques should be integrated into the teaching and learning of programming courses rather than using either one of them in its entirety. Arguably, by complementing the activities of both techniques, students could learn programming more effectively such that they could acquire all the components of the CT skills. Certainly, more studies are needed to focus on the impact of the integration of problem-solving and metacognitive techniques on the development of a strong CT skill among programming students. This study is a part of an ongoing major study consisting of several phases. The ensuing part of the study would focus on the development of a teaching and learning model that could serve as a guideline to enhance students' CT skills based on experts' opinions.

7. Acknowledgment

The author wishes to extend her appreciation to the Ministry of Education of Malaysia for the research grant [FRGS/1/2018/SS109/UPSI/02/29] which has led to the funding of this research and to the UPSI Research Management and Innovation Center for the assistance they have offered.

8. References

Abdullah, A. H., Rahman, S. N. S. A., & Hamzah, M. H. (2017). Metacognitive skills of Malaysian students in non-routine mathematical problem solving. *Bolema, Rio Claro (SP)*, *31*(57), 310 – 322.

Aizyl, A. (2016, August 11). PM: Schools lessons to integrate computational thinking from next year. https://www.malaymail.com/news/malaysia/2016/08/11/schools-to-integrate-computational-thinking-into-lessons-from-next-year-say/1181159

Barefootcas, Computational thinking. (2014). https://barefootcas.org.uk/barefoot-primary-computing resources/concepts/computational-thinking/.

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In *International Computing Education Research (ICER)*. Proceedings of the International Computing Research Education Workshop, (pp. 81-86). Seattle WA USA. https://doi.org/10.1145/1089786.1089794

Brennen, K., & Resnick, M. (2012). *New framework for studying and assessing the development of computational thinking* [Paper presentation at meeting]. American Educational Research Association, Vancouver, BC, Canada.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, *2*(2), 141-178. https://doi.org/10.1207/s15327809jls0202_2

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834-860. https://doi.org/10.3102/0034654317710096

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33-39. https://doi.org/10.1145/2998438

Djambong, T., & Freiman, V. (2016). Task-based assessment of students' computational thinking skills developed through visual programming or tangible coding environments. In *International Association for Development of the Information Society*. Proceedings of the International Conference on Cognition and Exploratory Learning in Digital Age (CELDA), (pp. 41-52). Mannheim, Germany.

Feyzi-Behnagh, R., Azevedo, R., Legowski, E., Reitmeyer, K., Tseytlin, E., & Crowley, R. S. (2014). Metacognitive scaffolds improve self-judgments of accuracy in a medical intelligent tutoring system. *Instructional Science, 42*(2), 159-181. http://dx.doi.org/10.1007%2Fs11251-013-9275-4

Filiz, K., Yasemin, G., & Volkan, K. (2016). A Framework for computational thinking based on a systematic research review. *Baltic J. Modern Computing*, *4*(3), 583-596.

Francom, G. M. (2010). Teach me how to learn: Principles for fostering students' self-directed learning skills. *International Journal of Self-Directed Learning, 7*(1), 29-44.

Gaeta, M., Mangione, G. R., Orciuoli, F., & Saverio, S. (2014). Metacognitive learning environment: a semantic perspective. *Journal of e-Learning and Knowledge Society*, *7*(2), 69-80. http://dx.doi.org/10.20368/1971-8829/522

Gill, T. G. (2011). *Informing with the case method: A Guide to case method research, writing, and facilitation*. Informing Science Press.

Halpern, D. F. (1999). Teaching for critical thinking: Helping college students develop the skills and dispositions of a critical thinker. *New Directions for Teaching and Learning*, *1999*(80), 69-74. https://doi.org/10.1002/tl.8005

Havenga, M. (2015). The role of metacognitive skills in solving object-oriented programming problems: A case study. *TD the Journal for Transdisciplinary Research in Southern Africa*, *11*(1), 133-147. http://dx.doi.org/10.4102/td.v11i1.36

Hinojosa, L. M. M., Rodriguez, M. C., & Paez, C. A. O. (2020). Measurement of metacognition: Adaptation of metacognitive state inventory in Spanish to Mexican University students. *European Journal of Educational Research, 9*(1), 413-421. http://dx.doi.org/10.12973/eu-jer.9.1.413

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S. J. (2015). A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. *Journal of Computer Assisted Learning,* 1–7. http://dx.doi.org/10.1111/jcal.12099

Joseph, L. M., Alber-Morgan, S., Cullen, J., & Rouse, C. (2016). The effects of self-questioning on reading comprehension: A literature review. *Reading & Writing Quarterly, 32*(2), 152 - 173. http://dx.doi.org/10.1080/10573569.2014.891449

Joseph, K. J. R. (2016, August 12). Computer science education to debut in schools next year. https://www.thestar.com.my/news/nation/2016/08/12/grooming-students-to-be-techsavvy-computer-science-education-to-debut-in-schools-next-year/

Kafai, Y. B. (2016). From computational thinking to computational participation in K--12 education. *Communications of the ACM*, 26–27. https://doi.org/10.1145/2955114

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K–8 curriculum. *ACM Inroads*, *5*(4), 64-71. https://doi.org/10.1145/2684721.2684736

Malaysia Digital Economy Corporation [MDEC]. (2018). Computational Thinking and Computer Science Teaching Certificate Programme for Educator. Ministry of Education Malaysia.

Malik, S. I., & Coldwell-Neilson, J. (2017). Impact of a new teaching and learning approach in an introductory programming course. *Journal of Educational Computing Research,* 1–31. http://dx.doi.org/10.1177/0735633116685852

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. In *Proceedings of the Working*

Group Reports of 2014 on Innovation & Technology in Computer Science Education Conference, ITiCSE-WGR (pp. 1-29). https://doi.org/10.1145/2713609.2713610

Margarida, R., Alexandre, L., & Benjamin, L. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education, 14,* 42. https://doi.org/10.1186/s41239-017-0080-z

Mayer, R. E. (2003). *Learning and instruction*. Prentice-Hall.

Ministry of Education Malaysia. (2012). Pelan pembangunan pendidikan Malaysia 2013-2025 [Malaysia education development plan 2013-2025]. Ministry of Education Malaysia.

Ministry of Education Malaysia. (2016). Kurikulum standard sekolah rendah KSSR. Bahagian Pembangunan Kurikulum [KSSR primary school standard curriculum. Curriculum Development Division]. Ministry of Education Malaysia.

Mohd Rum, S. N. (2015). A metacognitive support environment for novice programmer using semantic web [Doctoral dissertation]. University of Malaya, Kuala Lumpur.

Nimmi, P. M., & Zakkariya, K. A. (2016). Developing metacognitive skills: A potential intervention for employability enhancement. *Journal of Contemporary Research in Management*, *11*(3), 11-20.

Nunaki, J. H., Damopolii, I., Kandowangko, N. Y., & Nusantari, E. (2019). The effectiveness of Inquiry-based learning to train the students' metacognitive skills based on gender differences. *International Journal of Instruction*, *12*(2), 505–516. https://doi.org/10.29333/iji.2019.12232a

Nurul Faeizah, H., Hairulliza, M. J., Siti Aishah, H., & Hazilah, M. A. (2020). Technology integration to promote desire to learn programming in higher education. *International Journal on Advanced Science, Engineering, and Information Technology*, *10*(1), 253-259. http://dx.doi.org/10.18517/ijaseit.10.1.10264

Nor Hasbiah, U., & Jamilah, H. (2019). A Web-based Learning programming portal: Do instructors need it to enhance novice students' computational thinking skills? *International Journal of Innovative Technology and Exploring Engineering*, 8(9), 1945-1958.

Park, N. (2016). Development of computer education program using LOGO programming and fractals learning for enhancing creativity: Focus on creative problem-solving. *International Journal of u- and e-Service, Science and Technology, 9*(2), 121–126. http://dx.doi.org/10.14257/ijunesst.2016.9.2.13

Paul, R., Fisher, A., & Nosich, G. (1993). Workshop on critical thinking strategies: Foundation for Critical Thinking. Sonoma State University, CA.

Poli, R., & Koza, J. (2014). *Genetic Programming*. Springer.

Ramdiah, S., & Duran Corebima, A. (2014). Learning strategy equalizing students' achievement, metacognitive, and critical thinking skills. *American Journal of Educational Research*, *2*(8) 577-584. http://dx.doi.org/10.12691/education-2-8-3

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities under-lie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behaviour, 72,* 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Selby, C. C. (2015). Relationships: computational thinking, pedagogy of programming, and bloom's taxonomy. In *Proceedings of the Workshop in Primary and Secondary Computing Education,* (pp. 80-87). https://doi.org/10.1145/2818314.2818315

Seong-Won, K., & Youngjun, L. (2020). An analysis of pre-service teachers' learning process in programming learning. *International Journal on Advanced Science, Engineering, and Information Technology*, *10*(1), 58-69. https://doi.org/10.18517/ijaseit.10.1.5723

Shamir, G., Tsybulsky, D., & Levin, L. (2019). Introducing computational thinking practices in learning science of elementary school. In *Proceedings of the Informing Science and Information Technology Education Conference*, (pp. 187-205). Jerusalem, Israel. https://doi.org/10.28945/4327

Soumela, A., & Stavros, D. (2014). How to support students' computational thinking skills in educational robotics activities. In *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*. Padova, Italy.

Sterberg, R. J., & Sternberg, K. (2012). *Cognition* (6th ed.). Cengage Learning.

Swaid, S. I. (2015). Science direct bringing computational thinking to STEM education. *Procedia Manufacturing, 3,* 3657–3662. https://doi.org/10.1016/j.promfg.2015.07.761

Ung, L. L., Tammie, C. S., Jane, L., & Norazila, A. A. (2017). Preliminary investigation: Teachers' perception on computational thinking concepts. *Journal of Telecommunication and Computer Engineering*, *9*, 2-9.

Uysal, M. P. (2014). Improving first computer programming experiences: The case of adapting a web-supported and well-structured problem-solving method to a traditional course. *Contemporary Educational* Technology, *5*(3), 198-217. https://doi.org/10.30935/cedtech/6125

Veenman, M. V. J., Van Hout-Wolters, B. H. A. M., & Afflerbach, P. (2006). Metacognition and learning: Conceptual and methodological considerations. *Metacognition Learning, 1*, 3-14.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society,* 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Xabier, B., Miguel, A. O., Juan, C. O., & Mauricio, J. R. (2018). Computational thinking in pre-university blended learning classrooms. *Computers in Human Behaviour, 80,* 412-419. https://doi.org/10.1016/j.chb.2017.04.058