Discovering the Role of Problem-Solving and Discussion Techniques in the Teaching Programming Environment to Improve Students' Computational Thinking Skills

Nor Hasbiah Ubaidullah, Zulkifley Mohamed, Jamilah Hamid, and Suliana Sulaiman

***Abstract*—Computational thinking skill is one of the essential abilities to be learned and perfected by students of this century. Studies have shown that in the teaching and learning of programming courses, discussion and problem-solving techniques have been widely used. However, studies based on the suitability of such teaching techniques for the development of the computational thinking skills of students are, however, lacking. In this context, this research was conducted to define the teaching techniques used by university lecturers when teaching a computer programming subject and to explore how the techniques can influence the development of the computational thinking skills of students. This research was based on a combination of qualitative and quantitative approaches involving a semi-structured interview and a survey method, respectively. The research sample consisted of eight (8) university lecturers recruited from several Malaysian public universities, who had been teaching computer science to undergraduates. The results showed that in teaching computer programming, a majority of the respondents used discussion and problem-solving methods, with each assisting students to gain computer programming skills and learn certain components of computational thinking. As such, it is recommended that teaching practitioners incorporate the discussion and problem-solving techniques in the teaching and learning of programming courses. The incorporation of such strategies will help students develop good computer programming and computational thinking skills encompassing all the fundamental elements. The results also revealed that the respondents had no experience in using the metacognitive technique. As such, it is also proposed that future research should focus on this technique to investigate any possible effects that it may have on the growth of the computer programming and computational thinking skills of undergraduates.**

***Index Terms*—Computational thinking skill, computer programming, problem-solving technique, discussion technique, metacognitive technique.**

I. Introduction

Computational thinking is one of the latest skills that is much needed for student learning in the 21st century, which has witnessed the emergence of a generation of digital producers rather than digital consumers. As such, many countries have revamped their education systems by integrating elements of this ability into their school curricula. Basically, computational thinking skill is an ability that enables a person to understand how an event will take place. To date, many concepts have been used to describe this capacity, each with its own unique features. Selby [1] highlighted that computational thinking skills consist of five components, namely abstraction, algorithm, decomposition, evaluation, and generalization. Computational thinking skills are recognized as essential for students to learn computer science and IT [2]. Such acknowledgement is evident from recent studies focusing on the development of this ability among university and college students through programming courses [3]–[7].

In higher education, computer programming is one of the core subjects in computer science and information technology, the learning of which has been found daunting [3], [8] because students need to learn and develop skills that include problem solving, critical thinking, and writing codes in a specific programming language. Apparently, most students often lack such skills to help them effectively learn computer science and IT courses.

Ideally, students should possess not only high cognitive capabilities but also sound metacognitive abilities, from which information and skills can be classified [9]. In principle, metacognitive knowledge refers to the kind of knowledge that helps students monitor their knowledge during the learning process. Metacognitive skills, on the other hand, include tasks such as preparation, monitoring, and assessment [10], [11]. As such, the teaching and learning of computer programming entail metacognitive skills to help students solve programming problems [12]. Students must also have sound problem-solving skills to help them analyze programming problems critically. Essentially, solving programming problems may involve several essential steps, namely the understanding and description of a programming problem, preparation, design, writing of codes, and testing, all of which can enhance computational thinking skills of students.

As such, Malaysia should emulate other countries in nurturing students with good computational thinking skills. To date, studies focusing on the development of such skills have been conducted by several Malaysian researchers. However, most of these studies were mainly focused on primary and secondary school children in learning computational thinking. Thus far, only a few studies have been devoted to examining the effects of computational

Manuscript received April 9, 2021; revised June 11, 2021. This research was conducted under FRGS [2019-0017-107-02] grant which is sponsored by Malaysia's Ministry of Education.

N. H. Ubaidullah, J. Hamid, and S. Sulaiman are with the Faculty of Arts, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Malaysia (Corresponding author: N. H. Ubaidullah; e-mail: hasbiah@fskik.upsi.edu.my, hjamilah@upsi.edu.my, suliana@fskik.upsi.edu.my).

Z. Mohamed is with the Faculty of Sciences and Mathematics, Universiti Pendidikan Sultan Idris, Perak Malaysia (e−mail: zulkifley@fsmt.upsi.edu.my).

thinking on students at the tertiary level [13]–[15]. Given a lack of such studies, it is therefore necessary for Malaysia's research community to pursue additional studies of this nature at the tertiary level. Premised in this context, this research was conducted with the following objectives: 1) To identify techniques or approaches used in teaching computer programming, 2) To investigate the adequacy of such techniques or approaches in improving undergraduates' programming and computational thinking skills, and 3) To examine lecturers' knowledge of computational thinking skills.

The following research questions were formulated to help answer the above objectives.

1) What techniques or approaches are used in the teaching of computer programming, and why are they deemed appropriate?
2) How can such techniques help students improve their programming and computational thinking skills?
3) Are lecturers well-versed in computational thinking skills?

II. Methodology

The research was based on a qualitative approach involving a semi-structured interview technique. This technique enables researchers to obtain data from research subjects or respondents that provide more insights of a particular phenomenon. Additional data were also elicited through a questionnaire that was distributed to the respondents to get their feedback on their knowledge of computational thinking skills.

A. Respondents

Eight (8) computer programming lecturers from several public universities in Malaysia were selected for this research. The lecturers were recruited based on several factors, including their experiences and engagement in the teaching of the course at the bachelor's degree level.

B. Instrument

The set of questions for the semi-structured interview was used as the key research instrument in this research. Questions were systematically organised by asking general questions first and then followed by posing more probing questions. In addition, the researchers had also developed an interview checklist to guide the posing of relevant questions to help elicit feedback on issues related to computational thinking skills and the techniques used in the teaching and learning of computer programming. Several qualitative analysis experts were recruited to validate the checklist and the set of questions.

C. Procedure

In this study, the research procedure was performed in three stages:

- *Stage 1*: To obtain written approvals from the deans of the relevant faculties to allow their computer science or IT lecturers to be interviewed.
- *Stage 2*: To conduct online discussions and determine the dates for the interviews of selected lecturers.
- *Stage 3*: To interview the selected lecturers on the pre-determined dates.

Each interview lasted for about 40 to 60 minutes and the transcriptions of all answers, opinions, and ideas were recorded.

D. Data Analysis

Data elicited from the interviews were analyzed using the thematic analysis technique that helped the researchers to do the following: 1) To understand and examine the data, 2) To arrange relevant codes or to encode the data, and 3) To determine appropriate themes (relevant to the research questions). In this study, two themes emerged from the recorded transcriptions, which were coded as Theme 1 and Theme 2. On the other hand, the quantitative data obtained from the questionnaire were analyzed descriptively.

III. Findings

The demographics of the eight (8) respondents, namely their academic positions, ages, and working experiences, are summarized in Table I. As shown, the respondents were made up of four senior lecturers and four lecturers, whose ages ranged from 40 to 55 years. The majority of the respondents taught the subject matter in the first semester of the academic year involving computer programming undergraduates in their respective institutions.

TABLE I: Respondents' Demography

No.	Current Position	Age (in years)	Teaching experience (in year)	Semester of study
1	Lecturer	40	10	1
2	Senior lecturer	49	24	1
3	Senior lecturer	50	25	2
4	Lecturer	42	15	1
5	Lecturer	47	18	1
6	Senior lecturer	52	27	1
7	Senior lecturer	55	29	2
8	Lecturer	45	15	1

The discussion of this section is divided into two themes, namely Theme 1 that concerns with the techniques used in the teaching and learning of computer programming and Theme 2 that deals with the computational thinking ability of the respondents.

A. Theme 1: The Techniques Used in the Teaching and Learning of Computer Programming

The detailed analysis of the techniques used in the teaching and learning of computer programming was based on three 'open-response' questions, which allowed the respondents to provide more comprehensive answers.

Question 1: Do you use a particular technique in teaching computer programming? If so, how do you apply such a technique?

In answering to the above question list, respondents R1 and R4 indicated that they had not applied any particular technique in teaching computer programming. On the other

hand, respondents R5 and R8 claimed that they did use some techniques, but no information was given to indicate the exact nature of the techniques used, as shown below:

R5 – "*I instructed my students on what they had to do to answer these questions, and together I evaluated their responses".*

R8 – "*Initially, I asked my students to read a question and then discussed the required inputs, processes, and output to solve such a question".*

Similarly, respondents R2, R3, R6, and R7 indicated that they used discussions and problem-solving methods to teach computer programming based on their answers as follows:

R2 – *"I had discussed and taught my students to think logically about solving a problem, helping them analyze the problem, creating algorithms and drawing up codes; and then writing computer codes on the computer".*

R3 – *"I taught my students to think critically about their assignments by assisting them to comprehend the issues, to identify the input, process, output, and suitable control structure necessary, and to draw up the flowcharts needed. Later, they developed the draft codes, which I then reviewed. Finally, the codes were run on the computer".*

R6 – *"I would generally address a problem with my students while at the same time urge them to provide solutions, analyse the problem, create main algorithms or pseudo-codes, and write draft codes on a piece of paper. This was crucial to ensure that the programming logics were appropriate before the coding was carried out on the computer".*

R7 – *"Planning was crucial to solving problems. Students had to plan what kind of input, process, and output was needed, including an acceptable control structure, such as 'if and while', and flowcharts. They would then continue to execute the coding on the computer".*

Question 2: Do you believe the technique you used had helped students improve their computer programming skills, thereby improving their computational thinking skills? Why is it a good idea to use it?

In response to this question, respondent R2 stated that discussion and problem-solving techniques were the best techniques for teaching computer programming, as indicated by his response as follows: "*In my teaching, I have used both discussion and problem-solving techniques because I noticed that such techniques have helped my students to improve their computer programming skills".*

Likewise, respondents R3, R6, and R7 emphasized the use of the problem-solving technique in their teaching of computer programming, as evidenced by their responses as follows:

R3 – *"Problem-solving, in my opinion, is the best technique for teaching computer programming because it consists of several steps to guide students in solving programming problems. Students will not lose track if they strictly follow such steps carefully, which ultimately and gradually help them to improve their programming skills".*

R6 – *"I would contend that problem-solving technique is the most appropriate technique for teaching and learning computer programming. This technique covers all the steps needed in solving computer problems. Thus, by following all the steps, students will be able to practice solving programming problems more efficaciously, which in turn helps them improve their programming skills".*

R7 – *"I have used problem-solving as a teaching technique, and I believe it is the best technique for teaching and learning computer programming. As students practice solving programming problems with the steps prescribed by this technique, they will be able to gain programming skills"*

Regarding the questions pertaining to the techniques that could help improve students' computational thinking, respondents R5, R6, and R8 indicated that they were uncertain as they were not familiar with the concept of computational thinking. Likewise, respondents R1 and R4 indicated that they were also unsure about such a concept as they had not used any specific technique in teaching computer programming. By contrast, respondents R2, R3, and R7 stated that they had some knowledge about computational thinking as they had used the problem-solving technique before when teaching computer programming (which was revealed during the interview for Question 1). Given that they had some familiarity with such a concept, they were confident that their teaching techniques had helped improve their students' computational thinking skills, which were made evidently clear by their responses as follows:

R2 - *"Given that the elements of computational thinking are relevant to problem-solving activities in programming, the use of such a technique is deemed highly effective in improving students' computational thinking skills".*

R3 – "*As students' computational thinking skills can be improved through the learning of computer programming, any specific techniques used in the teaching of programming can indirectly improve their computational thinking skills as well".*

R7 – "*Many studies have shown that computational thinking skills can be developed through the learning of computer programming. As such, I am of the opinion that the problem-solving technique is highly practical to help improve students' computational thinking skills".*

Question 3: Have you ever heard the term 'metacognitive' and do you believe it is necessary to use the metacognitive technique in teaching computer programming to help develop students' computer programming and computational thinking skills? How will this technique improve both skills?

In response to the question above, most respondents claimed that they had heard such a term, but they were not entirely clear of its practical significance for the development of computational thinking. In particular, respondents P3's and P7's responses were quite insightful to shed some light onto the practicality of such a technique, as indicated as

follows:

R3 – *"I have no extensive expertise in the use of the metacognitive technique, but it may definitely contribute to the development of students' computational thinking skills if it is used appropriately in learning programming. Obviously, we need to carry out more studies on the use of this technique in the teaching and learning of computer programming, particularly by focusing on how it can improve computer programming and CT skills".*

R7 – *"Metacognitive is, in my view, a widely used term in education. Given that I come from the science background, I have little knowledge about it. However, I did read some papers suggesting that the metacognitive technique has also been used in teaching computer programming. As such, more efforts are entailed to conduct more studies to explore the efficacy of the use of such a technique in the teaching and learning process that can help enhance students' computer programming and computational thinking skills".*

Table II summarizes the results of respondents' responses to questions based on the first theme (Theme 1). As highlighted, the discussion and problem-solving techniques were the two teaching techniques mainly used by the respondents, indicating that such teaching techniques are widely used in Malaysia's public universities by programming lecturers. By contrast, none of the respondents had ever used the metacognitive technique in their teaching practice.

It was also revealed that most of the respondents were unsure of the appropriateness of the use of problem-solving technique to enhance students' computational skills, with the exception of respondents R2, R3 and R7 who provided the 'not sure' responses. Similarly, most of the respondents were uncertain about the appropriateness of the use of the metacognitive technique for the development of students' computational skill, as highlighted by their 'not sure' responses. Only respondents R3 and R7 indicated that such a technique was appropriate, as demonstrated by their 'appropriate' responses.

B. Theme 2: Respondents' Knowledge of Computational Thinking

Table III summarises the results of the survey which focused on the respondent's insights of computational thinking by answering six 'yes-or-no' questions. As shown, all the respondents were familiar with the term 'computational thinking' based on their responses to Question 1. Evidently, they had some degree of knowledge of such a term and believed that students could develop their computational thinking skills through the learning of programming courses. In particular, their answers to questions Q2 and Q3 revealed not only their ability to correctly list the elements of computational thinking but also their sound understanding of the meaning of the term. It was also observed that none of the respondents had any experiences in teaching computational thinking courses, as reflected by their answers to question Q5. Likewise, almost all of them, except for respondent R7, indicated that they had never undertaken any kind of research on computational thinking based on their responses to question Q6.

TABLE II: THE SUMMARY OF THE FINDINGS BASED ON THE FIRST THEME

Respondent	Technique used in teaching computer programming			The appropriateness of the problem-solving technique for CP and CT skills acquisition	The appropriateness of the metacognitive technique for CP and CT skills acquisition
	Discussion technique	Problem-solving technique	Metacognitive technique		
R1	√	x	x	Not sure	Not sure
R2	√	√ (analysis, design, and coding)	x	Appropriate	Not sure
R3	x	√ (analysis, planning, and design)	x	Appropriate	Appropriate
R4	√	x	x	Not sure	Not sure
R5	√	√ (analysis)	x	Not sure	Not sure
R6	√	√ (analysis, design, and coding)	x	Not sure	Not sure
R7	x	√ (analysis, planning, design, and coding)	x	Highly appropriate	Appropriate
R8	√	√ (analysis)	x	Not sure	Not sure

TABLE III: RESPONDENTS' KNOWLEDGE ON CT

Questions/ Respondents	R1	R2	R3	R4	R5	R6	R7	R8
1. Are you familiar with the term computational thinking?	√	√	√	√	√	√	√	√
2. Do you understand the meaning of the term computational thinking?	√	√	√	√	×	×	√	×
3. Do you know the elements of computational thinking?								
- Abstraction	×	√	√	√	×	×	√	×
- Algorithm thinking	√	√	√	×	×	×	√	×
- Decomposition	×	√	√	√	×	×	√	×
- Generalization	×	×	×	×	×	×	×	×
- Evaluation	×	×	×	×	×	×	×	×
4. Do you agree that programming courses can support and develop students' computational thinking skills?	√	√	√	√	√	√	√	√
5. Have you ever been involved in teaching computational thinking courses?	×	×	×	×	×	×	×	×
6. Have you ever been involved in studies/ research on computational thinking?	×	×	×	×	×	×	√	×

IV. DISCUSSION

The discussion of this section is divided into two subsections as follows:

A. Discussion Based on Theme 1

The first theme (Theme 1) deals with the existing computer programming techniques used by the respondents and their views on the appropriateness of such techniques to enhance students' computer programming and computational thinking skills. Table II summarizes the results of their responses to questions based on this theme. As shown, six respondents noted that they had used the discussion and problem-solving method in their teaching practices.

The discussions of the findings are as follows:

1). Discussion *technique:* Certainly, discussions play an important role in the teaching and learning sessions to stimulate students to think logically that helps them solve a given problem. Through discussions, programming students, especially those with few experiences, will be able to envision some ideas on how to tackle a given problem. Despite the pervasive use of such a technique, it lacks concrete steps to help tackle programming problems. Arguably, the efficacy of the discussion technique relies on the ingenuity and experience of lecturers. As indicated, respondents R1, R2, R4, R5, R6, and R8 mainly used discussions to help their students solve programming problems. The pervasive use of such a technique was not surprising given that it is one of the two common techniques, including the problem-solving technique, widely used by most lecturers in teaching programming courses [16].

2) Problem-Solving technique: Problem-solving is a common approach used by most lecturers in teaching programming courses. Primarily, students use the computer to solve problems when learning programming -- a technique that is distinct from that of solving mathematical problems without using the computer. The steps of solving computer programming problems are primarily based on computer outputs. By following the steps of the problem-solving technique, students will be able to build and strengthen their computer programming abilities and computational thinking skills.

- Comprehension and description of the problem.

In this first step, students were given a question that entailed them to identify a corresponding problem. Based on the requirements of a given problem, they would be able to examine the critical input, process, and output required to solve the problem. As shown in Table II, respondents R2, R3, R5, R6, R7, and R8 analyzed the problem with their students that helped reveal the necessary input, process, and output in computer programming. Admittedly, such an analysis plays an integral part in leaning computer programming.

Students would be able to develop skills related to the abstraction aspect of computational thinking through this analytical method, which enables them to abstract relevant data associated with a given problem and eliminate unrelated or redundant data [17]. Thus, the first step in problem-solving of computer programming can help students improve their computer programming and abstraction skills.

- Planning problem-solving.

Planning is another important aspect of computer programming that requires students to apply their knowledge to plan a number of ways to solve a given problem and to prioritize the one which is considered the most appropriate. In this regard, respondent R7's answer was highly revealing, which underscored the significance of the second phase in problem-solving as highlighted by the following response: "*Planning is essential in solving a problem. To define the necessary structure, such as 'if' and 'while', as well as to prepare a suitable flowchart, students need to plan the necessary input, process, and output. Then, on a sheet of paper, they have to perform the coding and later apply the coding on the computer.*" Given this revelation, it becomes imperative that this second step of problem-solving, namely planning, be embedded in the teaching of computer programming, the effect of which can help students develop strong programming skills.

Likewise, students need to break down a large, complicated problem into several manageable sub-problems, which help them to develop two elements of computational thinking. The first will assist students to move the problem-solving process to a new problem that they are

familiar with [4], [18]. The second, on the other hand, will help students dissect the problems into smaller components that can be easily solved [19].

- Designing problem-solving.

The results of the interviews showed that half of respondents (namely R2, R3, R6, and R7) strongly emphasized the third stage of problem-solving, namely the design of problem-solving, in their teaching practices. Designing is one of the essential activities after the planning process in computer programming. Typically, students are required to write a few algorithms centered on the analysis and planning, which were performed earlier in the designing activities, and to establish the logical soundness of such algorithms such that they correspond well with an output. Therefore, the implementation of designing activities in the problem-solving technique can help students improve their computer programming skills, particularly in designing. Such a practice was consistent with the activities carried out by [20], [21]. Primarily, these tasks represent the algorithmic thinking of computational thinking among students. Arguably, students may indirectly acquire this sub-skill of computational thinking by participating in designing activities in computer programming.

- Writing codes.

Coding is another important activity in computer programming in which students write relevant codes using a particular programming language. In other words, students need to perform such an activity to produce an output for a given programming problem. Also, they have to check any errors, such as logical and syntactical errors, in a program. Therefore, performing such an activity entails students to have sound logical thinking. Similarly, creativity plays an important role in generating accurate, reliable computer outputs. As revealed, respondents R2, R3, R6, and R7 insisted that their students had to write draft codes on a piece of paper before they performed the coding process on the computer. This, effectively, helped students to improve their computer programming skills in terms coding and algorithmic thinking skills. Given that coding is an integral part of learning programming courses, students are expected to develop sound algorithmic thinking by engaging in coding activities, which have been found to be highly effective in helping inexperienced students to learn the subject matter [16].

- Testing

Programming testing can be carried out iteratively by using a number of different inputs. Such iterations are necessary to ensure that the outputs produced are accurate. The practical significance of such iterative testing, however, was properly understood by the respondents, as none of them gave any indications to acknowledge its importance. Given this finding, it is important that students be informed of the significance of testing, which is indispensable in their learning practice to produce highly reliable computer outputs. By implementing testing in problem-solving, they will be able improve their computer programming skills in terms of testing. In fact, programming testing activities are consistent with the principle of concepts rooted in computational thinking that is to ensure the soundness of system solutions and algorithms to achieve the intended objectives [9]. To date, several studies have been devoted to examining the impacts of testing activities on the development of students' computer programming skills [16], [20], [21].

3) Metacognitive technique: As shown in Table II, the results of the interviews revealed that most respondents had neither used the metacognitive technique in their teaching activities nor acknowledged its importance for the development of students' computational thinking. The few exceptions were respondents R3 and R7, who did suggest that such a technique was necessary in their teaching practice to enhance students' computer programming skills. Similarly, most of the respondents, with the exception of respondents R3 and R7, were unaware of the efficacy of the metacognitive technique. The former argued that if such a technique was acceptable for use in teaching computer programming, then it would certainly be acceptable to use it as a means to enhance students' computational thinking skills. On the other hand, the latter admitted that such a technique was highly acceptable, an admission that was hardly surprising as he seemed to have a reasonably strong understanding of computational thinking in view of his experiences in conducting computational thinking studies. Thus, such findings indicate that lecturers' experiences have a strong impact on shaping their opinions regarding the suitability of the metacognitive technique as an effective method to enhance student's computational thinking skills, as affirmed by [12].

The results of this study of metacognitive technique are not consistent with those of previous studies such as [12] and [22], in which the former revealed that the use of such a technique had a significant effect on student learning. Admittedly, conflicting results might have been attributed to several factors, including experiential factor. For example, in this study, the respondents were university lecturers who were from science background who normally may not be familiar with the concept of metacognition or metacognitive technique. By contrast, those from the social science background, such as school teachers, may have a greater understanding of such a concept or computational thinking. Given this stark contrast, it is reasonable to argue that the background of research subjects may give rise to different results. Interestingly, one of the respondents, namely respondent R7, proposed that further research should be conducted on such a technique to explore its impacts on the development of students' computational thinking skills, suggesting that there may be some academicians who are aware of the importance of the metacognitive technique.

Typically, in learning computer programming, students must have the skills to enable them to discern a given programming problem logically and critically such as to gain a full understanding of what needs to be done. Also, they need to determine the precise problem-solving process, apply appropriate programming strategies, eliminate programming errors, assess their problem-solving approaches, and test the programming outputs. Surely, such problem-solving steps may entail the use of the metacognitive technique to teach students to plan problem-solving strategies, monitor their

execution, and assess their effectiveness. In other words, students must manage the programming process, articulate their actions, and seek alternative solutions to improve their programming skills. Therein lies the need for using the metacognitive technique in the learning of computer programming, which can indirectly help improve students' programming skills, as has been amply demonstrated in a study by [12].

The above discussions on the three teaching techniques help highlight the following revelations:

- Respondents who had been teaching computer programming used the discussion technique quite successfully that helped their students to think logically and critically. To a certain extent, the use of this technique managed to strengthen their students' computational thinking skills.
- To teach computer programming to their students, the respondents also used the problem-solving technique, which is a technique commonly used by most teaching practitioners. Typically, solving a given programming problem involves several basic steps that students have to strictly follow, with each step comprising an action that pertains to a specific component of computational thinking. As such, the use of the problem-solving technique in the teaching of computer programming can indirectly help improve students' computational thinking skills.
- In their instructional activities, the respondents did not employ the metacognitive technique. Two of the respondents' responses, however, were quite insightful in that they remarked such a technique was practically feasible to be used not only in the teaching of computer programming but also as an effective means to develop and nurture students' computational thinking skills [12].

The discussions that focus on the first theme (Theme 1) allow the researchers to answer the first and second research questions that pertain to the teaching methods used by respondents in teaching programming and to how such techniques have helped them to enhance their students' programming and computational thinking skills, respectively.

B. Discussion Based on Theme 2

Computational thinking skill is one of the important skills required in this information-driven century that students must acquire. Apparently, many teaching practitioners seem to recognize that the use of such a skill is only confined computer-related fields. In fact, many are unaware that computational thinking is also needed in other fields of knowledge, such as mathematics and science.

Recent studies have shown that most university lecturers teaching computer science have little knowledge of such a skill. These results are not consistent with the findings from an earlier study [16], which found pre-university lecturers interviewed had a thorough understanding of computational thinking, which they had embedded in the teaching of programming courses. Similarly, [2] found that school teachers had sound knowledge of computational thinking, which enabled them to embed such a cognitive concept in their teaching practices. Such conflicting findings between university lecturers and school teachers may be attributed to the latter's greater exposure to such a concept, as they might have attended several training courses related to computational thinking, which have been made mandatory by the Ministry of Education of Malaysia in response to the infusion of the elements of computational thinking in the new primary and secondary school curricula [13].

Given the above findings, it is imperative for university lecturers to equip themselves with sound knowledge of computational thinking to ensure they can train their students to develop such a skill, which is surely needed in studies at the tertiary level. It was also revealed that there is a lack of studies on computational thinking skills carried out by lecturers (as shown in Table III). As such, it is important for them to attend training courses in computational thinking that can help them develop relevant knowledge and expertise which can help guide their future undertakings in teaching and researching of such a skill. For example, university lecturers can carry out more studies that focus on the appropriate techniques of teaching and learning to enhance students' computational thinking skills. The scope of such studies should be broadened beyond the field of computer science to include other educational fields, such science and mathematics. By pursuing such efforts, they can certainly gain a greater insight of the elements of computational thinking that they can capitalize on to improve and innovate their teaching practices. The call for such efforts is best exemplified by the respondent R7's responses, stressing that teaching practitioners need to undertake more studies of computational thinking to help them gain a firm grasp of such an important concept in the twenty-first century. Overall, the above discussions help the researchers to answer the third research question that delves into the current knowledge of computational thinking among university lecturers.

V. Conclusion

Arguably, learning computer programming courses is very challenging, especially to novice students. To make matters worse, they must demonstrate their abilities or computer programming skills to develop and run accurate, efficient computer programs at the end of such courses. Thus, programming instructors must use appropriate teaching and learning techniques to help improve students' computer programming skills. This research revealed that most university lecturers have been relying on the discussion and problem-solving techniques to teach computer programming to computer science undergraduates. Such revelation is hardly surprising as the discussion technique can help instructors to stimulate students' logical and critical thinking needed in solving programming problems. Likewise, the problem-solving technique has also been widely used by lecturers to teach the same courses. Essentially, this technique consists of several steps that students must follow to solve a given programming problem. By strictly following such steps in a proper sequence, students will be able solve the problem, ultimately helping them to develop strong computational thinking, which is one of the important skills needed in learning in the twenty-first century. It was also observed that even though most lecturers have some knowledge of such a cognitive concept, but it does not

account much. Hence, they need to attend more training courses to help them gain more knowledge of computational thinking, which they can embed in their teaching practices to help students develop such a skill.

The findings of this research provide strong evidence to suggest that the discussion and problem-solving techniques are highly efficacious in helping students not only to improve strong computer programming skills but also to develop sound computational thinking skills as well. As such, it is recommended that teaching practitioners incorporate these two techniques in the teaching and learning of programming course, the incorporation of which will help students learn all elements of such skills, which leads to the development of strong computer programming skills and computational thinking skills. The findings also revealed that the use of the metacognitive techniques in the teaching and learning of computer programming has not been extensively explored. Thus, more efforts are entailed to carry out studies on its potential impacts on the development of computer science undergraduates' computer programming and computational thinking skills. The findings of this research, which are based on an ongoing study, will serve as an important input for the development of the next phase of the same study.

Conflict of Interest

The authors declare that there is no conflict of interest in the undertaking of this study.

Author Contributions

Associate Professor Dr. Nor Hasbiah Ubaidullah is the principal researcher of this study, assisted by Dr. Jamilah Hamid, Dr. Suliana Sulaiman, and Associate Professor Dr. Zulkifley Mohamed as co-researchers. This paper was written by Associate Professor Dr. Nor Hasbiah Ubaidullah and Associate Professor Dr. Zulkifley Mohamed and the final version was approved by all authors.

Acknowledgment

We wish to express our gratitude to Malaysia's Ministry of Education for the Fundamental Research Grant Scheme [2019-0017-107-02] that helped fund this research and to RMIC of UPSI for the support rendered throughout this research.

References

[1] C. Selby, "Relationships: Computational thinking, pedagogy of programming, and bloom's taxonomy," in *Proc. the Workshop in Primary and Secondary Computing Education on ZZZ*, pp. 80–87, 2015.

[2] S. Aslina, "Developing students' computational thinking skill through cooperative learning based on hands-on, inquiry-based, and student-centric learning approaches," in *Proc. Conference 2nd International Teacher Education Conference on Teaching Practice (ITE Computational ThinkingP 2018)*, 2018.

[3] S.-W. Kim and Y. Lee, "An analysis of pre-service teachers' learning process in programming learning," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 10, no. 1, pp. 58-69, 2020.

[4] B. Xabier, A. O. Miguel, C. O. Juan, and J. R. Mauricio, "Computational thinking in pre-university blended learning classrooms," *Computers in Human Behavior*, vol. 80, 2018.

[5] R. Margarida, L. Alexandre, and L. Benjamin, "Computational thinking development through creative programming in higher education," *International Journal of Educational Technology in Higher Education*, vol. 14, no. 42, 2017.

[6] T. Djambong and V. Freiman, "Task-based assessment of students' computational thinking skills developed through visual programming or tangible coding environments," in *Proc. 13th International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2016)*, pp. 41–52, 2016.

[7] K. Cagin, K. Mary, B. Liz, and M. Lachlan, "A serious game for developing computational thinking and learning introductory computer programming," *Procedia - Social and Behavioral Sciences*, vol. 47, pp. 1991 – 1999, 2012.

[8] H. Nurul Faeizah, M. J. Hairulliza, H. Siti Aishah Hanawi, and M. A. Hazilah, "Technology integration to promote desire to learn programming in higher education," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 10, no. 1, pp. 253-259, 2020.

[9] A. H. Abdullah, S. N. S. A. Rahman, and M. H. Hamzah, "Metacognitive skills of Malaysian students in non-routine mathematical problem solving," *Bolema, Rio Claro (SP)*, vol. 31, no. 57, pp. 310 – 322, 2017.

[10] M. Gaeta, G. R. Mangione, F. Orciuoli, and S. Saverio, "Metacognitive learning environment: A semantic perspective," *Journal of e-Learning and Knowledge Society*, vol.7, no. 2, pp. 69-80, 2014.

[11] P.M. Nimmi and K.A. Zakkariya, "Developing metacognitive skills: A potential intervention for employability enhancement," *Journal of Contemporary Research in Management*, vol. 11, no. 3, pp. 11-20, 2016.

[12] M. Havenga, "The role of metacognitive skills in solving object-oriented programming problems: a case study," *TD the Journal for Transdiciplinary Research in Southern Africa*, vol. 11, no. 1, pp. 133-147, 2015.

[13] L. L. Ung, C. S. Tammie, L. Jane, and A. A. Norazila, "Preliminary investigation: teachers' perception on computational thinking concepts," *Journal of Telecommunication and Computer Engineering*, vol. 9, pp. 2-9, 2017.

[14] *Ministry of Education Malaysia, Kurikulum Standard Sekolah Rendah KSSR. B. P. Kurikulum*, Ed. Ministry of Education Malaysia, 2016.

[15] *Ministry of Education Malaysia, Pelan Pembangunan Pendidikan Malaysia 2013-2025*, Ministry of Education Malaysia, 2012.

[16] U. N. Hasbiah and H. Jamilah, "A web-based learning programming portal: Do instructors need it to enhance novice students' computational thinking skill?" *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no, 9, 1945-1958, 2019.

[17] *Malaysia Digital Economy Corporation (MDEC)*, Computational Thinking and Computer Science Teaching Certificate Programme for Educator, 2018.

[18] A. Soumela and D. Stavros, "How to support students' computational thinking skills in educational robotics activities," in *Proc. 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*, 2014.

[19] G. Shamir, D. Tsybulsky, and L. Levin, "Introducing computational thinking practices in learning science of elementary school," in *Proc. the Informing Science and Information Technology Education Conference*, 2019, pp. 187-205, 2019.

[20] S. I. Malik and J. Coldwell-Neilson, "Impact of a new teaching and learning approach in an introductory programming course," *Journal of Educational Computing Research*, pp. 1–31, 2017.

[21] D. Hooshyar, R. B. Ahmad, M. Yousefi, F. D. Yusop, and S. J. Horng, "A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers," *Journal of Computer Assisted Learning*, pp. 1–7, 2015.

[22] M. R. S. Nurulain. "A metacognitive support environment for novice programmer using semantic web," PhD Thesis, Universiti of Malaya, Kuala Lumpur, 2015.

Nor Hasbiah Ubaidullah is an associate professor in the Faculty of Arts, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Malaysia. She obtained a bachelor of science degree in computer science and PhD in information technology from Universiti Kebangsaan Malaysia. She earned her master of science degree in information systems from

the University of Salford, UK. Since 1991, she has been actively involved in the teaching of computer science and related courses at several institutions of higher learning. Her research is primarily focused on educational technology, educational software, and computer science education. Currently, she teaches programming and engineering courseware courses and is actively involved in the research area of computational thinking and programming. Also, she has been appointed by the Malaysian Qualification Agency (MQA) as an assessor for the IT course in the bachelor's degree and master's degree programs offered by various public and private universities in Malaysia.

Zulkifley Mohamed is an associate professor in the Faculty of Sciences and Mathematics, Universiti Pendidikan Sultan Idris, Malaysia. He earned a bachelor of science degree in statistics from MARA University of Technology Malaysia. In 1997, the University of Salford, UK, awarded him a master's degree in applied statistics and operational research. He complete d his PhD in statistics from Universiti Kebangsaan Malaysia. He has a vast experience in the teaching of statistics and related courses at the tertiary level of education. He has conducted many outstanding research works, notably robust statistics, statistical modelling, and mathematics education. He has also been appointed by the Malaysian Qualification Agency (MQA) as an assessor for the Statistics course in the bachelor's degree and master's degree programs offered by various public and private universities in Malaysia.

Jamilah Hamid is a senior lecturer in the Faculty of Arts, Computing, and Creative Industry, Universiti Pendidikan Sultan Idris. She holds a bachelor of science degree in education (hons.) majoring in mathematics and master of science (information technology). She obtained her PhD in internet and web computing from Universiti Pendidikan Sultan Idris, Malaysia. She has been teaching in several institutions of higher learning since 1991, teaching several courses, such as programming, methodology in courseware development, educational technology, and teaching methodology. She has a keen interest in virtual reality, educational technology, educational software, and teaching methodology in programming. Currently, she teaches the methodology in teaching methodology and assessment course at Universiti Pendidikan Sultan Idris.

Suliana Sulaiman is a senior lecturer in the Faculty of Arts, Computing, and Creative Industry, Universiti Pendidikan Sultan Idris. She holds a bachelors' degree in computer science (artificial intelligence) from the University of Malaya, Malaysia. Later, she earned a master of science degree and a PhD in science and information system from Universiti Kebangsaan Malaysia, Malaysia. She started her teaching career in higher education in 2005 and has been teaching several courses, such as principles of software engineering, engineering courseware, software engineering process, audio-video technique, and games programming. Her research interests delve into natural language processing, recommender system, intelligent educational system, and educational technology.