Analysis on the Requirements of Computational Thinking Skills to Overcome the Difficulties in Learning Programming

Karimah Mohd Yusoff[1], Noraidah Sahari Ashaari[2], Tengku Siti Meriam Tengku Wook[3], Noorazeen Mohd Ali[4]
Matriculation Division, Ministry of Education Malaysia, Putrajaya, Malaysia[1]
Software Technology and Management System, Universiti Kebangsaan Malaysia, Bangi, Selangor, Malaysia[2, 3, 4]

***Abstract*—Programming has evolved as an effort to strengthen science, technology, engineering and mathematics (STEM). Programming is a complex process, especially for novices, since it requires problem-solving skills to solve problems of developing algorithms and programme codes. Problem-solving competencies, which are necessary as 21st-century skills, include a set of cognitive skills that are related to problem-solving and programme development or specifically known as computational thinking (CT) skills. In particular, this study quantitatively assessed the computational thinking skills in the context of programming, specifically on the difficulties in learning programming. From the perspectives of the instructors, the survey results highlighted the need to implement CT skills as an approach in teaching and learning programming. A model for teaching and learning programming is necessary as a guide for instructors in the teaching and learning process of programming.**

Keywords—Problem-solving; STEM; difficulties in learning programming; cognitive; novice

I. Introduction

Job opportunities and daily activities that involve computers have encouraged students to pursue computing, such as computer engineering, computer science, information science, and software engineering, as a career [1]. The U.S. Bureau of Labour Statistics reported that computing represented 71% of the careers in science, technology, engineering and mathematics (STEM) by 2018 [2]. The programming curriculum has gained growing attention given the significance of computing in meeting the current needs and STEM agenda. Programming is emphasized in schools and even at the pre-university level in order to provide students with a good knowledge base and programming skills. However, it is a challenging and complex process to learn programming [3][4][1], since it requires good cognitive ability. Novice programmers often face problems during the introductory course of programming [5][6] that may cause hesitation to pursue the advanced courses of programming. This scenario shows that the early mastery of programming serves as a catalyst for students to consider courses related to programming at a later stage.

In general, programming is viewed as a means of producing computer programmes. Basically, programming solves real-world problems through computer programmes. The implementation of the identified solution involves several steps, which are as follows: (1) formulate a problem; (2) design a solution by generating an algorithm; (3) translate the algorithm into a programme code; (4) test and evaluate the complete programme. Although studies have introduced several programming methods and approaches to assist novice programmers, not all focus on the programming steps involved in solving problems. Furthermore, most of the past studies focused on mastering the concepts of programming, such as learning using MicroWorlds, game-based learning, story-based learning, and visualisation tools.

For instance, the use of MicroWorlds, such as Alice, Greenfoot, Marine Biology Case Study, Scratch, and Turtle Graphics, provides a user-based interface (GUI) that introduces basic programming concepts to novices. Although such approach can build problem-solving skills, it mainly focuses on implementing solutions in specific programming languages. In addition, it does not develop one's ability to formulate problems, design solutions, and generate algorithms that are often necessary for large, complex problems or involve multiple processes. Besides, games-based learning approaches are considered to provide students with a lot of fun while learning, but it also focusses on programming concepts [7]. Meanwhile, work-based learning approaches involve problem solving in programming that can help to reduce the learners' cognitive load by performing solutions based on work examples [8][9][10]. Most importantly, the learning process must involve learners themselves, where the learning process is designed according to their competencies. Learners are also required to engage in a group discussion to discuss, interact and provide feedback, and guide their peers.

Accordingly, the idea of computational thinking (CT) already existed during the early 1950s and has gained growing attention of educators and researchers over the past decade. The term "computational thinking" was first used by Seymour Papert in 1980 and 1996 [11][12]. Following that, Wing [13] formally introduced CT as an approach to solving problems, designing systems, and understanding human behaviour, which reflects the basic concepts of computing. The discussion of CT in the literature is often associated with problem-solving [14].

CT is an important and necessary way of thinking for computer programmers and other professionals in STEM [15]. CT skills, which include decomposition, abstraction, pattern-recognition, algorithm, logical reasoning, and assessment (or evaluation) skills, are cognitive skills that can be used in the teaching and learning process of programming that typically involves problem-solving. The computational skills of CT derived from computer science have the potential to be used for problem solving in all disciplines [16]. Through these computational skills, one can be better at solving problems and can identify problems and apply a smart approach to solve the problems [17]. Problem solving in programming involves several processes that can be implemented using appropriate CT skills. In general, there are two main phases of problem-solving in programming. The first phase is to generate an algorithm whereas the second phase is to develop a programme. Both phases coincide with the role of CT as a thought process that involves formulating problems and solutions in a form that can be effectively implemented by the information processing agents [18] such as computers.

Although CT is an ideal teaching and learning approach that can help with the curriculum problems [19], its implementation, to date, focuses on K-12 only [20][21][22][23]. Studies have revealed limited CT implementation for higher learning, particularly at the pre-university level. Practical research on teaching CT skills at the higher education level has been continuously implemented in computer science and STEM [17]. The difficulties in learning programming have been a topic of discussion among educators and researchers.

The difficulties in learning programming were widely explored in past studies [24][25][26][27][28][29][30][31]. Besides that, Du Boulay [3], Robins, Rountree, and Rountree [4], and Qian and Lehman [32] also reviewed the difficulties in learning programming. Some of the identified difficulties included designing solution plans, developing algorithms, syntactic mastery, writing and evaluating programmes, cognitive requirements, and limited programming ability. The problem-solving approach can be implemented using CT skills according to the required role. For instance, abstraction skills play a role in identifying and retrieving relevant information to determine key ideas and reduce unnecessary information. Besides that, decomposition skills help to decompose complex problems (that involve several processes) according to the process, which makes problem solving easier, as the problems can now be solved in parts. Meanwhile, through pattern-recognition skills, programmers can observe the patterns, trends, and regularity of data by observing the similarities and differences with other problems. There are also the algorithm skills that involve a set of rules and instructions to execute tasks or address any problem-solving needs in programming. These skills can help programmers to develop computer algorithms, specifically the step-by-step solutions into forms that can be implemented by a computer. Apart from that, there are logical reasoning skills by analysing and studying facts based on accurate and clear-thinking approach. After all, problem-solving involves logic. Last but not least, the needs of each solution should be evaluated, which highlights the important role of evaluation (or assessment) skills in determining the adequacy of an algorithm, system, or process in serving its purpose of meeting the needs.

In short, CT plays an important role in learning and solving problems and computerizes thinking in all disciplines [33] given its significance as the core of STEM for solving problems and designing large, complex systems [12]. Focusing on the significance of 21st-century skills, learners need to master CT skills in order to solve problems based on the principles of computer science [14]. Clearly, CT is a thinking approach to develop problem-solving skills using the basic concepts of computing. In view of the above, the current study focused on high-level approaches to solve problems in programming using CT proficiency.

Due to the nature of programming that closely related with the problem-solving, computational thinking skills are potential to overcome the difficulties in programming. This study reviewed related literature on the difficulties in learning programming, how these difficulties are linked to computational thinking, and the need for computational thinking in learning programming. The obtained findings of this study were expected to benefit both instructors and students or novice programmers, especially in the preparation of an effective teaching and learning approach to programming. Hence, in this paper, author concerns to study the needs of computational thinking skills to overcome the difficulties in learning programming.

II. Research Purpose

For more than a decade, the use of CT as an approach to problem-based learning has prompted researchers to explore its use in computer science, especially for programming [34][35][36][37]. In addition, there is an increasing need to understand the role and skills of CT and identify the need to use CT skills in problem-solving and programme development. With that, this study aimed to identify the difficulties in learning programming and the required CT skills among learners in order to facilitate the teaching and learning process of programming. In particular, this study addressed the following research questions:

1) What are the common difficulties in learning programming among students?

2) What are the CT skills that are associated with the identified difficulties in learning programming?

3) Do the instructors face the identified difficulties?

4) What are the CT skills needed to overcome the difficulties in learning programming?

III. Methodology

A. *Mapping the Difficulties in Learning Programming with Computational Thinking Skills*

With respect to the purpose of this study, the difficulties in learning programming, especially among novice programmers, were reviewed. For this study, the difficulties in learning programming were first mapped based on the review of key literature, analysis of documents, and the elements of difficulties in relation to the CT skills. Fig. 1 illustrates the applied mapping method.

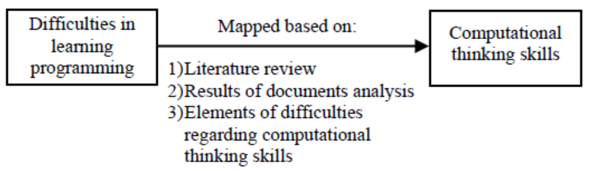
Difficulties in learning programming
Mapped based on:
1)Literature review
2)Results of documents analysis
3)Elements of difficulties regarding computational thinking skills
Computational thinking skills

Fig. 1. Mapping Method for Difficulties in Learning Programming with Computational Thinking Skills.

Meanwhile, Table I presents the mapping results, which showed that the difficulties in learning programming are entirely related to all CT skills, such as abstraction, decomposition, pattern-recognition, algorithms, logical reasoning, and assessment (or evaluation) skills. These initial findings demonstrated the significance of CT skills in developing the required instrument for the ensuing survey for this study.

TABLE I. RELATIONSHIP OF DIFFICULTIES IN LEARNING PROGRAMMING WITH COMPUTATIONAL THINKING SKILLS

References	***Difficulties in programming***	***Justifications of mapping to computational thinking (CT) skills***	***Computational Thinking (CT) Skills***					
			Decomposition	***Abstraction***	***Pattern-Recognition***	***Algorithm***	***Logical Reasoning***	***Assessment or Evaluation***
Qian and Lehman [32]	1) Not familiar with the syntax	1) Syntax is closely related to the closeness of mapping, which is a relationship between the programming languages and students' existing knowledge of the concepts used. The existing knowledge refers to knowledge in programming or other knowledge that applies the same concepts to programming. It is related to the pattern-recognition skills. 2) The existing knowledge used to predict is part of logical reasoning [38]. 3) Evaluation skills are indirectly required when logical reasoning is used to predict.			√		√	√
	2) Lack of ability in mathematics	1) The ability in mathematics is related to cognitive skills. 2) CT skills are a set of cognitive skills. These skills are in tandem with problem-solving skills for mathematics. 3) Problem-solving for mathematics requires strategy; perform solution sequentially; and involve logical reasoning and evaluation skills.	√	√	√	√	√	√
	3) Lack of mental model to implement codes	1) This is related to the concepts of programming. It involves cognitive skills to master it. 2) The use of analogies in daily life can help to understand the implementation of codes, as variables can hold one value at a time; each statement ends with a semicolon and each repeated process has a numerator to track it. This is related to pattern-recognition, logical reasoning, and evaluation skills. 3) Students in the study were found to face problem to understand how codes work. This issue is related to algorithm since logic in programming is in line with the logic flow of algorithm.			√	√	√	√
	4) Lack of strategic knowledge	1) This leads to difficulties in designing solutions, writing programmes, and resolving errors. 2) These issues are related to decomposition, abstraction, pattern-recognition, algorithm, logical reasoning, and evaluation skills.	√	√	√	√	√	√
	5) Lack of existing knowledge in programming	1) The existing knowledge is related to pattern-recognition, logical thinking, and assessment skills. 2) Existing knowledge that is used to predict is part of logical reasoning [38].			√		√	√

References	***Difficulties in programming***	***Justifications of mapping to computational thinking (CT) skills***	***Computational Thinking (CT) Skills***					
			Decomposition	***Abstraction***	***Pattern-Recognition***	***Algorithm***	***Logical Reasoning***	***Assessment or Evaluation***
	6) Difficult to solve complex tasks	1) Complex tasks involve students' cognitive load. 2) It is related to strategies of problem solving with respect to decomposition skill to decompose complex problems, abstraction skill to simplify the problem by removing unimportant information while not losing important information, and the ability to adapt other methods of almost similar solutions. 3) Students in the study were found to demonstrate the tendency of making mistakes when it comes to managing complex tasks. 4) Involves logical thinking and assessment skills.	√	√	√	√	√	√
Kwon [31]	7) Difficulty in designing a solution plan	1) Some of the proposed strategies are: a) Decompose the complex task in parts b) Retrieve relevant information to perform solution c) Identify other similar problem and adapt its solution 2) Abstraction and algorithm are closely related to the capability of problem solving [39]. 3) Students in the study were found to fully understand the problem and have the ability to describe the solution but difficult to figure out a solution or develop instructions that can be implemented by a computer. Related to the ability to generate or develop algorithms. Logical thinking skill is also involved in developing algorithms.	√	√	√	√	√	
Papadopoulos and Tegos [29]	8) Lack of problem solving	1) Problem solving involves planning such in problem (7) to perform solution in the form of algorithm.	√	√	√	√	√	
	9) Lack of CT skills	2) Indirectly related to CT skills	√	√	√	√	√	√
Siti Rosminah and Ahmad Zamzuri [30]	10) Difficult to understand the basic concepts of programming structures and programme design	1) Based on the literature, this is related to the existing knowledge. This problem can be solved using the phenomenon in daily life as a comparison to understand the concepts of programming structure. It is related to pattern recognition skill. 2) Designing programmes is related to the ability to develop an algorithm or symbolic languages that describe solutions in programme codes. 3) Logical reasoning is used to predict the aftermath of recognising the patterns and generate the algorithms. Logical reasoning assesses whether the algorithm is correct and meets its purpose. 4) The evaluation ensures that the design of the programme is good and meets its purpose [40].			√	√	√	√
	11) Difficult to master the syntax of programming language	1) Students need to understand what they want to achieve and the process to achieve results. For example: a) The instructions for text display to specifically display the "Hello" text.			√		√	√

References	***Difficulties in programming***	***Justifications of mapping to computational thinking (CT) skills***	***Computational Thinking (CT) Skills***					
			Decomposition	***Abstraction***	***Pattern-Recognition***	***Algorithm***	***Logical Reasoning***	***Assessment or Evaluation***
		b) The instructions to enter inputs like scores. 2) Syntax is related to the programming language structure. Connecting with other concepts can assist students to learn syntax, such as a sentence must end with a period (.), while in programming, a statement ends with a semicolon (;). This is related to pattern-recognition skill. 3) Logical reasoning and evaluation skills help students to improve the ability to master syntax.						
	12) Difficult to understand the abstract concepts that involve the position of variables in computer memory	1) Analogies in daily life can help to understand the abstract concepts, such as an object with its content refers to the variable that holds its value. 2) The abstract concepts can also be conveyed using teaching aids, such as visualisation. For example, in computer memory, there is a variable that holds a value; the input entered would be held by the variable.			√		√	
Chan Mow [28]	13) Cognitive needs	1) CT skills refer to the cognitive process in problem solving [37].	√	√	√	√	√	√
Renumol, Jayaprakash, and Janakiram [41]	14) Cognitive difficulties	Same as the above problem (13)	√	√	√	√	√	√
Haberman and Muller [42]	15) Difficult to use the abstraction process	1) If problems are complex, decomposition skill is required to break down the problem into smaller parts; so, it would be easier to manage. Decomposition is a prerequisite for abstraction [14] when it comes to complex problems. The abstraction process to retrieve relevant information for each section is made after decomposition. 2) It is directly related to the need for abstraction skill. 3) Pattern-recognition skill can be helpful because the pattern-oriented instruction approach influences abstraction skill [27].	√	√	√			
Gomes & Mendes [26]	16) Cannot write a programme and develop an algorithm	1) Abstraction, decomposition, and pattern-recognition skills are best used as strategies to design solutions for the development of algorithms. It also requires logical thinking skill because an algorithm is a series of steps in the form that can be processed by a computer. 2) The study suggested emphasising the development of problem-solving skills among the novices. 3) Decomposition, abstraction, pattern-recognition, and algorithm skills are part of problem-solving skills. 4) Logical thinking is required to develop algorithms.	√	√	√	√	√	
Lahtinen, Ala-Mutka, and Jarvinen [25]	17) Difficult to understand the concepts	1) The study found the involvement of complex cognitive to understand without the phenomenon in daily life for			√	√	√	

References	***Difficulties in programming***	***Justifications of mapping to computational thinking (CT) skills***	***Computational Thinking (CT) Skills***					
			Decomposition	***Abstraction***	***Pattern-Recognition***	***Algorithm***	***Logical Reasoning***	***Assessment or Evaluation***
	of programming related to recursion, instruction, and abstract data	comparison. The study proposed appropriate design of teaching and learning materials to help students to master the concepts of programming through the comparison of concepts in life. Pattern recognition skill can help the students to master the concepts of programming that include: a) Recursive functions can be demonstrated through the concept of reuse, such as factorial or other situations in life. b) Demonstrates the use of pointers through the phenomena in daily life that can represent the concepts of programming c) Use other representations in daily life to illustrate abstract data 2) The concepts of programming are related to algorithm skill. 3) Logical thinking is necessary to master the concepts of programming.						
	18) Difficult to develop programmes	1) Materials such as the instructions to convert the algorithms to programme codes are used as references for students. Students can refer to the examples to write the programme. It is related to pattern-recognition skill. 2) The development of programmes involves syntax and semantics. Students must understand the flow of algorithms and use logical thinking skill to develop and evaluate programmes.			√	√	√	
Robins, Rountree, and Rountree [4]	19) Lack of strategy to plan the solutions and design algorithms	1) Decomposition, abstraction, pattern-recognition, and algorithm skills are important problem-solving skills for programming. Students need to think logically to design the algorithms.	√	√	√	√	√	
	20) Implement algorithms and write programmes	1) A guide to translate algorithms into suitable programme codes as a reference for students to implement algorithms. Approaches such as reusing, modifying, or integrating the existing programmes are strategies to develop new programmes. It is related to pattern-recognition skill. 2) Students must understand the flow of algorithms before writing the programme to ensure that the solution design is logical and valid. 3) The development and evaluation of programmes require logical thinking and assessment skills.			√	√	√	√
	21) Evaluate programmes and tracking and fixing errors	1) Require logical reasoning to evaluate and debug the programme					√	√
Winslow [24]	22) Problem to combine syntax and semantics to	1) Syntax is related to the structure of programming languages whereas semantics relates to the logic or concepts of statements, expressions, or			√	√	√	√

References	***Difficulties in programming***	***Justifications of mapping to computational thinking (CT) skills***	***Computational Thinking (CT) Skills***					
			Decomposition	***Abstraction***	***Pattern-Recognition***	***Algorithm***	***Logical Reasoning***	***Assessment or Evaluation***
	produce a valid programme	programmes. 2) Students need to understand the flow of algorithms before writing the program because an algorithm is a symbolic language that describes the solution in the form of programme code. It deals with algorithm and logical reasoning skills. 3) Syntax relates to the closeness of mapping, which is a relationship of programming languages with the students' existing knowledge of the concepts used. It is related to pattern recognition. 4) The semantics relates to the logic or concept for statement, expression, or programme. Students need to evaluate whether the algorithm or programme is logic and meets its purpose. Evaluation skill is used to ensure that the programme performs well and is able to achieve its goal [40]. Relates to logical reasoning and evaluation skills.						
Du Boulay [3]	23) Cognitive needs of programming	1) CT skills refer to cognitive processes in problem-solving [37]. Six CT skills are deemed very relevant to support problem solving in programming.	√	√	√	√	√	√
	24) Syntax and semantics	1) Syntax refers to the programming language structure. It is related to the closeness of mapping, which is the relationship of programming language with the students' existing knowledge of the problem or the concept that students want to learn. This is related to pattern-recognition skill. 2) The semantics relates to the logic or concept for statement, expression, or program are logic and valid. This stage uses logical reasoning and evaluation skills to evaluate an algorithm or programme.			√	√	√	√
	25) Lack of support skills (pragmatic)	1) Pragmatics refers to the practical aspects of how language features can be used to achieve multiple objectives. This is related to the logical reasoning skills. 2) Evaluation skill is also required to determine whether the programme is written in line with its objectives. The study then concluded that students need to learn the skills of how to determine, develop, test, and debug using the available tools.					√	√
	26) Orientation	1) The need to understand its uses, the types of problems that can be solved, and its advantages in programming 2) It is difficult for students to identify the terms of the programme, the actual process needed, and its usefulness. Students can be guided by programming-oriented contexts used in real life. It is directly related to pattern-recognition, logical reasoning and evaluation skills.			√		√	√

Overall, the difficulties in learning programming can be categorised as cognitive difficulties, difficulties in designing solutions, difficulties in developing algorithms, difficulties in writing and evaluating programmes, difficulties in combining syntax and semantics, difficulties related to the concepts of programming, and limited programming skills. The cognitive difficulties are related to difficulties in completing complex tasks, cognitive programming needs, lack of ability in mathematics, and less computational thinking. With limited cognitive abilities, it would also be difficult to design solutions. Difficulties in designing solutions refer to difficulties in using the abstraction process and designing solutions. As a result, it would be difficult to develop an algorithm as well.

Besides that, there are also the lack of strategies to design algorithms, difficult to understand the concepts of programming structure and programme design. Such scenario would inevitably lead to difficulties in writing and evaluating programmes. There are also other lack of strategies in implementing algorithms, difficulties in combining syntax and semantics (to produce a complete programme), and difficulties in evaluating programmes, debugging error, and tracking and correcting the errors. Moreover, learners who are not familiar with syntax would not be able to master semantics. Apart from the strategies to plan and develop algorithms, understanding the concepts of programming, which are typically related to recursive, pointer, abstract data, and mental models to implement programs, is also important. Last but not least, there are also the lack of support skills and orientation difficulties when it comes to the difficulties in learning programming.

B. *Mapping Validation by Experts*

These initial findings were then validated by the appointed experts. In this case, the mapping validation by experts referred to the expert evaluation on the CT skills in relation to the difficulties in learning programming. The mapping method in Fig. 1 was applied. The tabulated justifications of mapping to CT skills in Table III were reviewed by the experts. These experts were required to indicate their agreement to the established categories in the table.

Overall, the experts agreed on the mapped CT skills. There were additional expert reviews provided. For instance, students who are capable in mathematics may still encounter difficulty in mastering the logic of programming. The experts argued that the difficulties to come up with algorithms and programs are related to logical reasoning skill instead. In addition, certain students may not be able to come up with the solution because they evaluate the problem as a single, whole problem, rather than assessing the problem in different stages or processes. Such scenario demonstrates the students' incapability in problem-solving strategies, such as formulating problems to understand and design solutions and relate them to the existing knowledge and experience.

C. *Conducting a Survey*

Following that, an instrument was developed. In general, the instrument included a list of statements on the difficulties in learning programming. Considering the purpose of this study, the survey items aimed to measure all six identified CT skills, which were abstraction, decomposition, pattern-recognition, algorithm, logical reasoning, and evaluation skills. The details of the items for each construct are presented in Table II.

A total of 17 items were developed for each construct. A five-point Likert scale was employed. Likewise, these items were verified by experts before the actual data collection. Table III presents the level of measurement scale to determine the mean score of each construct.

A survey that involved instructors was then conducted to gather empirical data on the students' difficulties in learning programming. This study focused on programming lecturers from the Matriculation Division, Ministry of Education Malaysia. A total of 32 respondents participated in the survey. The survey aimed to identify the difficulties in learning programming based on the perspectives of these instructors. Through this survey, the need for CT skills to overcome the difficulties in learning programming can be identified to serve as a guide for the instructors in the teaching and learning process in programming.

TABLE II. DETAILS OF ITEMS FOR EACH CONSTRUCT

Construct	***Description of items***
Abstraction	Items were intended to identify the students' difficulties in understanding and formulating problems as well as identifying relevant information.
Decomposition	Items were intended to identify students' difficulties to decompose a problem that involves several processes in parts, so that it can be solved by section, as part of the problem-solving strategies.
Pattern-recognition	Items were intended to identify students' difficulties in integrating the existing knowledge and experience as problem-solving strategies.
Algorithm	Items were intended to identify students' difficulties in developing algorithms as well as their consequences if fail to create the algorithm.
Logical reasoning	Items were intended to identify students' difficulties in thinking logically by identifying and explaining the reasons behind the solution.
Evaluation	Items were intended to identify students' difficulties in evaluating the solution, whether the solution is suitable and meets its purpose.

TABLE III. LEVEL OF MEASUREMENT SCALE BASED ON MEAN SCORE

Mean score	***Level***
1.00 – 2.33	Low
2.34 – 3.67	Average
3.68 – 5.00	High

(Source: Landell, 1977) [43]

IV. ANALYSIS AND FINDINGS

There are two analysis and findings in this study. First, mapping the difficulties in learning programming with computational thinking skills. Second, survey among instructors on the students' difficulties in learning programming as well as the needs of CT skills. For the first analysis, the results of mapping help to identify the CT skills which is relevant to the difficulties in learning programming.

Some of the comments from experts described the problems encountered in programming learning. These results are useful for instructors to understand students' need in learning programming, helps to plan teaching and learning approaches or strategies and also in planning teaching materials and exercises.

Survey among instructors aim to identify students' difficulties in learning programming that in tandem with the needs of CT skills. Due to this purpose, data were analysed descriptively to determine the mean score of each item and construct, specifically to identify the level of need for each skill to overcome the difficulties in learning programming. As shown in Table IV, all items and constructs recorded high mean scores. The obtained results demonstrated that the difficulties in learning programming that were identified from literature are issues for learners. In addition, the results indirectly demonstrated the need for CT in teaching and learning programming. Algorithm skill recorded the highest mean score. This skill is useful for designing solutions by creating algorithms before writing the programs. Failure to develop algorithms may cause difficulty while coding. In addition, decomposition skill also showed high mean scores. This construct refers to the problems in managing large and complex problems. Due to this situation, it may cause problem to develop the algorithms. Based on these results, students need more problem solving practices to design the solutions. Hence, instructor must expose students with vary type of problems that require several processes to raise the strategies in problem solving. In conclusion, both of analysis and findings are useful to plan the strategy and approach in teaching and learning as well as to overcome the difficulties that commonly faced by students in programming.

TABLE IV. MEAN SCORE OF NEED FOR COMPUTATIONAL THINKING SKILLS IN TEACHING AND LEARNING PROGRAMMING

Construct	***Item***	***Mean score of items***	***Mean score of constructs***
Pattern-recognition	1	4.00	4.03
	2	4.06	
	3	4.03	
Decomposition	4	4.03	4.30
	5	4.34	
	6	4.53	
Abstraction	7	3.90	4.17
	8	4.22	
	9	4.38	
Algorithm	10	4.31	4.38
	11	4.41	
	12	4.41	
Logical reasoning	13	3.84	3.87
	14	3.78	
	15	4.03	
	16	3.75	
	17	3.94	
Evaluation	This item is contained indirectly in other constructs.		

V. CONCLUSION

Programming requires cognitive ability and involves strategies in planning and solving problems. Focusing on that, this study aimed to examine the difficulties in learning programming among students and determine the need for CT skills among instructors. This study first reviewed 11 empirical papers and three review papers on the difficulties in learning programming. Most of the identified difficulties in the past studies were related to cognitive needs, ability to plan solutions, difficulties in developing algorithms, and difficulties in writing and evaluating program. These identified difficulties were then mapped to the appropriate CT skills, which were validated by the experts. Following that, the items for each construct were developed for the survey. The survey specifically involved 32 instructors to gather empirical data on the difficulties in learning programming among students at the pre-university level. Based on the survey results, the identified difficulties in learning programming are clear among students today. Additionally, this directly demonstrated the need for CT skills in the teaching and learning process of programming. CT skills with appropriate approach or activities should be applied to guide students through real problems. The outcomes of mapping and survey were expected to contribute to the design of the problem-solving model and strategies in programming using CT skills, which can serve as a guide for instructors.

ACKNOWLEDGMENT

This study was funded by the UKM Research Grant (GUP-2018-155). We would also like to acknowledge the Matriculation Division, Ministry of Education Malaysia for their assistance and contribution for this study.

REFERENCES

[1] Vassilev, T. I. 2015. An Approach to Teaching Introductory Programming for IT Professionals Using Games. International Journal of Human Capital and Information Technology Professionals 6(1): 26–38.

[2] Stuikys, V. and Burbaite, R., 2018. Smart STEM-Driven Computer Science Education: Theory, Methodology and Robot-based Practices. Springer.

[3] Du Boulay, B. 1986. Some Difficulties of Learning to Program. Journal of Educational Computing Research 2(1): 57–73.

[4] Robins, A., Rountree, J. & Rountree, N. 2003. Learning and Teaching Programming: A Review and Discussion. Computer Science Education 13(2): 137–172.

[5] Yassine, A., Chenouni, D., Berrada, M. & Tahiri, A. 2017. International journal of emerging technologies in learning. International Journal of Emerging Technologies in Learning (iJET) 12(03): 110–127. Retrieved from http://online-journals.org/index.php/i-jet/article/view/6476.

[6] Watson, C. and Li, F.W., 2014, June. Failure rates in introductory programming revisited. In Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 39-44).

[7] Ibrahim, R., Rahim, N.Z.A., Ten, D.W.H., Yusoff, R., Maarop, N. and Yaacob, S., 2018. Student" s Opinions on Online Educational Games for Learning Programming Introductory. International Journal of Advanced Computer Science and Applications, 9(6), pp.332-340.

[8] Vieira, C., Yan, J. and Magana, A.J., 2015. Exploring design characteristics of worked examples to support programming and algorithm design. Journal of Computational Science Education, 6(1), pp.2-15.

[9] Margulieux, L.E. and Catrambone, R., 2016. Improving problem solving with subgoal labels in expository text and worked examples. Learning and Instruction, 42, pp.58-71.

[10] Jalani, N. H. & Sern, L. C. 2015. The Example-Problem-Based Learning Model: Applying Cognitive Load Theory. Procedia - Social and Behavioral Sciences 195: 872–880.

[11] Papert, S. 1980. Mindstorms; Children, Computers and Powerful Ideas. New York: Basic Book.

[12] Papert, S. and Harel, I., 1991. Situating constructionism. Constructionism, 36(2), pp.1-11.

[13] Wing, J. M. 2006. Computational thinking. Communications of the ACM 49(3): 33.

[14] Selby, C. 2015. Relationships: Computational Thinking, Pedagogy of Programming, and Bloom's Taxonomy. Proceedings of the Workshop in Primary and Secondary Computing Education 80–87.

[15] Estapa, A., Hutchison, A. & Nadolny, L. 2018. Recommendations to support computational thinking in the elementary classroom. International Technology and Engineering Educators Association. Retrieved from https://www.iteea.org/File.aspx?id=123563&v=25610bf

[16] Yadav, A., Gretter, S., Good, J. & Mclean, T. 2017. Computational Thinking in Teacher Education (November).

[17] Czerkawski, B. C. & Lyman, E. W. 2015. Exploring Issues About Computational Thinking in Higher Education. TechTrends 59(2): 57–65.

[18] Wing, J. M. 2010. Computational Thinking: What and Why? (November): 1–6.

[19] Shute, V. J., Sun, C. & Asbell-Clarke, J. 2017. Demystifying computational thinking. Educational Research Review 22: 142–158.

[20] Yadav, A., Hong, H. and Stephenson, C., 2016. Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. TechTrends, 60(6), pp.565-568.

[21] Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P. & Punie, Y. 2016. Developing Computational Thinking: Approaches and Orientations in K-12 Education. Proceedings EdMedia 2016.

[22] Gretter, S. & Yadav, A. 2016. Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills. TechTrends 60(5): 510–516.

[23] Kong, S. 2016. A framework of curriculum design for computational thinking development in K-12 education. Journal of Computers in Education 3(4): 377–394.

[24] Winslow, L. E. 1996. Programming Pedagogy --A Psychological Overview. ACM SIGCSE Bulletin 28(3): 17–22.

[25] Lahtinen, E., Ala-Mutka, K. & Jarvinen, H.-M. 2005. A study of the difficulties of novice programmers. ACM SIGCSE Bulletin 37(3): 14.

[26] Gomes, A. & Mendes, A. J. N. 2007. Learning to program-difficulties and solutions. International Conference on Engineering Education 1–5. Retrieved from http://ineer.org/Events/ICEE2007/papers/411.pdf

[27] Muller, O. & Haberman, B. 2008. Supporting abstraction processes in problem solving through pattern-oriented instruction. Computer Science Education 18(788840272): 187–212.

[28] Chan Mow, I. T. 2008. Issues and difficulties in teaching novice computer programming. Innovative Techniques in Instruction Technology, E-Learning, E-Assessment, and Education 199–204.

[29] Papadopoulos, Y. & Tegos, S. 2012. Using microworlds to introduce programming to novices. Proceedings of the 2012 16th Panhellenic Conference on Informatics, PCI 2012 180–185.

[30] Siti Rosminah, M. D. & Ahmad Zamzuri, M. A. 2012. Difficulties in learning programming: Views of students. 1st International Conference on Current Issues in Education (ICCIE 2012) (SEPTEMBER 2012): 74–79.

[31] Kwon, K. 2017. Student's misconception of programming reflected on problem-solving plans. International Journal of Computer Science Education in Schools 1(4): 14.

[32] Qian, Y. & Lehman, J. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming. ACM Transactions on Computing Education 18(1): 1–24.

[33] Bundy, A. 2007. Edinburgh Research Explorer Computational Thinking is Pervasive Computational Thinking is Pervasive 1(2): 2–5.

[34] Witherspoon, E.B., Higashi, R.M., Schunn, C.D., Baehr, E.C. and Shoop, R., 2017. Developing computational thinking through a virtual robotics programming curriculum. ACM Transactions on Computing Education (TOCE), 18(1), pp.1-20.

[35] Lye, S. Y. & Koh, J. H. L. 2014. Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior 41: 51–61.

[36] Lopez, A. R. & Garcia-Penalvo, F. J. 2016. Relationship of knowledge to learn in programming methodology and evaluation of computational thinking. Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM '16 73–77.

[37] Roman-Gonzalez, M., Perez-Gonzalez, J. C. & Jimenez-Fernandez, C. 2017. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior 72: 678–691.

[38] Barefoot Computing. t.th. Computational thinking. http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/.

[39] de Araujo, A. L. S. O., Andrade, W. L. & Guerrero, D. D. S. 2016. A Systematic Mapping Study on Assessing Computational Thinking Abilities. 2016 Ieee Frontiers in Education Conference (Fie) 1–9.

[40] Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. and Woollard, J., 2015. Computational thinking-A guide for teachers.

[41] Renumol, V., Jayaprakash, S. and Janakiram, D., 2009. Classification of cognitive difficulties of students to learn computer programming. Indian Institute of Technology, India, 12.

[42] Haberman, B. & Muller, O. 2008. Teaching abstraction to novices: Pattern-based and ADT-based problem-solving processes. Proceedings - Frontiers in Education Conference, FIE 7–12.

[43] Landell, K., 1997. Management by menu. London: Wilay and Sms Inc.