Improving College Students' Learning Performance in Computer Programming: The Case of Using the Polya Model

Jamilah Hamid
Saira Banu Omar Khan
Zafirah Mohd Adnan
Ummu Salmah Mohamad Hussin

DOI: https://doi.org/10.37178/ca-c.21.5.081

Jamilah Hamid,*Department of Computer, Faculty of Art, Computing & Creative Industry,Sultan Idris Education University, Tg Malim, Perak, Malaysia.*
Email: hjamilah@fskik.upsi.edu.my

Saira Banu Omar Khan, *Department of Computer, Faculty of Art, Computing & Creative Industry,Sultan Idris Education University, Tg Malim, Perak, Malaysia.*
Email: sairabanu@fskik.upsi.edu.my

Zafirah Mohd Adnan, *Department of Computer, Faculty of Art, Computing & Creative Industry,Sultan Idris Education University, Tg Malim, Perak, Malaysia.*
Email: zafirahkmj@gmail.com

Ummu Salmah Mohamad Hussin, *Information Technology Department, Sultan Azlan Shah University, Kuala Kangsar, Perak.*
Email: dr_ummu@usas.edu.my

Abstract

Many studies have shown that students with poor problem-solving skills are prone to failing programming courses. To help mitigate this problem, several problem-solving methods have been recommended, including the Polya Model. Therefore, this study aims to determine the impact of this model on matriculation college students' performances in learning programming algorithms. A quasi-experimental study with pre-test post-test research design was conducted to assess the performances in programming algorithm among a group of 30 college students, who were selected using the random sampling method. They were assigned into an experimental group and a control group who underwent an intervention treatment that lasted for seven weeks on two groups of students: control and treatment. The experimental and control groups learned programming algorithms using the Polya Model and conventional method, respectively. The research instrument used for the pre-test and post-test consisted of six subjective questions with three levels of difficulty based on Bloom's taxonomy. Descriptive statistics and a series of t-tests were used to analyze the data. The results of the t-test showed that there was a significant difference in the mean scores of the post-test measurements between the two groups, which favored the experimental group, especially for questions with a higher level of difficulty. This finding suggests that the Polya Model is an effective problem-solving technique that programming students need to learn to help improve their performances in learning programming algorithms at the collegiate level.

Keyword: Programming Algorithms, Programming, Polya Model, teaching and learning, HOTS, problem-solving skills.

INTRODUCTION

Admittedly, the need for highly skilled and knowledgeable workforce has become so urgent to help nations face the challenges in the highly competitive world. Therein lies the imperative for nations to improve their educational systems, starting from the elementary level upwards, to help students acquire the necessary skills in the twenty-first century. In Malaysia, several efforts have been put in place to improve school curricula to help students develop sound critical, creative, and innovative thinking skills, leadership skills, and strong ethical values. To achieve this aspiration, the Malaysian government has launched the Malaysian Education Blueprint[1], which aims to improve the quality of Science, Technology, Engineering and Mathematics (STEM) education by helping students to develop strong higher-order thinking skills (HOTS). To ensure the continuity of STEM education at the post-secondary level, several Malaysian matriculation colleges have conducted the Computer Science course in their Mathematical Science program, with programming as one of the major topics that students need to learn.

However, learning programming is considered a difficult, challenging task for most students who are new to programming [2, 3]. In programming, students need strong problem-solving skills are to help them learn the phases of algorithm development, the failure of which can be detrimental to their passing of the course. A report of the final Computer Science matriculation examination results for three consecutive years in one of the Malaysian matriculations centers shows that many students performed poorly due to a lack of problem-solving skills. For example, for the semester of 2017/2018, the examiners reported that students' skills in logic and programming were very low, evidenced by their failures to identify the essential elements in problem solving based on their wrong answers to questions regarding flow charts and programming. Therefore, the examiners suggested that problem-solving skills should be emphasized in the teaching and learning process. Likewise, the two semesters of 2016/2017 academic session saw similar results, with only 10 percent of the students (consisting of 644 candidates) scored full marks and were able to answer the programming algorithm questions correctly [4-6]signifying that they lacked the skills in developing programming algorithms. As such, these findings reinforce the need to improve matriculation college students' problem-solving skills before they continue their study at the university level.

Arguably, in programming, students need to have strong thinking skills to help them develop reliable working algorithms. Thinking skills are essential in learning process to form and understand concepts, engage in problem-solving, reasoning and creating or developing solutions. In this regard, the implementation of higher order thinking skills (HOTS) in the learning process helps enhance students' thinking ability, which is badly needed in today's competitive technology-driven world. [7]. assert that HOTS should be applied in the teaching and learning process, especially at the higher education level. Likewise, research conducted by [8] concluded that problem-solving skills are most important skills that emphasized by the programming lecturers or instructors and asserts that problem solving approaches in teaching computer programming needs a special approach to be used as a pedagogy. In a nutshell, the mastery of these skills is important to ensuring students can learn and understand the basic concepts of programming.

A pilot study was conducted to get some feedback on students' learning issues from 10 computer science lecturers of a matriculation college. The lecturers noted that the problem-solving steps prescribed in the Computer Science Curriculum Specification [9] lacked depth and, to make matters worse, such steps were discussed only in the lecture sessions. Some lecturers even claimed that they did not emphasize the problem-solving process explicitly, making students to ignore it their learning practices. Further compounding the problem was that lecturers used different teaching and learning methods based on their expertise and experience. However, all lecturers

agreed that students should have strong problem-solving skills to help them learn programming, especially for the topic of algorithm development.

As acknowledged, there is a myriad of factors affecting student learning of programming, including pedagogical strategies and teaching and learning methods [10-12]. As such, new, innovative methods are needed to help facilitate the teaching of programming, including the algorithm topic. Given that developing algorithms involves problem-solving, it is imperative for teachers and lecturers to use problem-solving models to help them teach such a difficult topic more effectively. For example [12]used a problem-solving model called ADRI model in their teaching programming. In addition, several studies have shown the use of several problem-solving models had significant impacts on students' achievements [13-17]

Among the models that commonly used in educational disciplines, notably in Mathematics education are Polya Model [18-21]. These models, however, have not been widely used in teaching programming at the tertiary level. Admittedly, this is a bit ironic given that the CSCS [20] clearly stipulates that problem-solving step are among the important concepts that programming students need to learn and master to ensure they can understand the basic processes of solving programming problems.

Based on the above discussion, there is a need to use problem-solving models to help improve existing teaching and learning practices such that students would be able to develop the essential skills that can them develop precise, efficient algorithms. Premised in this context, the researchers conducted this study to examine at the effects of using the Polya Model in the teaching and learning of programming algorithms on matriculation students' learning performances. The research objectives to guide this study are as follows:

1. To identify the elements in every phase of the adaptation of the Polya Model that are deemed suitable for the teaching and learning of programming algorithms.
2. To examine whether there was a significant difference in students' learning performance of programming algorithms between students who learned using the adaptation of Polya Model and those who learned using a conventional method.

Correspondingly, two research questions were formulated to address the above research objectives as follows:

1. Are the elements identified in every phase of the Polya Model suitable for the teaching and learning of programming algorithms?
2. Is there a significant difference in students' learning performance of programming algorithms between students who learned using the adaptation of Polya Model and those who learned using the conventional method?

To answer the second research question, the researchers formulated a research hypothesis as follows: There is no significant difference in the mean scores of students' learning performance in programming algorithms between students who learned using the adaptation of Polya Model and those who learned using the conventional method.

LITERATURE REVIEW

Learning programming algorithms

In principle, the learning process is a way in which a student focuses on processing, interpreting, and acquiring new information, knowledge, and skills [22] which varies depending on levels and contexts of learning. Thus, the learning process at the collegiate level is different from that at the school level [23]. In matriculation colleges, learning programming is implemented using the lecture method, complemented with classroom tutorials to enhance students' understanding of contents learned in lectures, the process of which creates a learning cycle [24]. The purpose of this cycle is to help students master algorithms, which is a foundation to solving problems and to encode programs as a formal representation of algorithms. However, a study conducted by [24] showed that most students were unable to master algorithms, indicating that the existing learning cycle was not effective to help achieve

the desired goals, which could be attributed to their lack of theoretical understanding and skills needed to learn algorithms.

According to [17], problem-solving skills can help students develop strong knowledge and improve their thinking skills. Admittedly, from the students' standpoint, problem-solving is the most difficult part in learning programming algorithms [12, 25]. Without these necessary skills, students will face some difficulties in learning computer programming, as they probably may not be able to understand programming questions. Hence, there is a compelling need to help students develop strong problem-solving skills using appropriate teaching approaches. In this regard, Yamashita et al. [24] assert the need to develop and use learning support systems to help students learn and understand programming algorithms more easily. Ideally, such systems should be compatible with the teaching and learning of programming algorithms that focus on problem solving through a systematic, effective learning process. They also argue that lecturers need to work harder to make the learning of programming algorithmic concepts more manageable and easier for students. Put simply, finding appropriate methods to help students learn and master the basics of algorithm development must be given a strong emphasis.

In Malaysian matriculation colleagues, lecturers teach computer programming based on the guidelines prescribed in the CSCS. However, such guidelines are quite general, providing no detailed information of the necessary steps and, thus, compelling lecturers to teach based on their own methods and understanding. The guideline consists of five (5) steps; (i) problem analysis, (ii) design a solution, (iii) implementation, (iv) testing and (v) documentation and for the algorithm development classes, the lecturers only focus until second steps. Reviewing the algorithm that has been designed (in step 2) which is very important neither is taught nor emphasize in the class for algorithm development topic. This practice runs counter to the recommendations of many scholars, such as [24] who stress the responsibility of lecturers to use appropriate teaching strategies to help students learn and understand algorithms more effectively.

Problem-solving skills and HOTS in learning programming

Thinking is the mental activity of the highest level that humans rely on to function effectively and efficiently. From the learning perspective, thinking skills involve a series of systematic processes that students perform in their learning activities [1]. According to [26], thinking involves scientific process which requires the mind to engage in series of sub-processes that can be divided into two categories: lower order cognitive process and high order thinking. This process is applied to any problem given, regardless the nature of problem. Admittedly, students have various levels of thinking skills that determine their levels of learning efficacy. In the educational realm, Bloom's taxonomy has been widely used to measure students' thinking levels. Essentially, this taxonomy consists of a set of hierarchical models used to measure the various learning levels that entail different skills that students need to acquire, as determined by their teachers. Bloom's taxonomy posits that learning at a higher level depends on the knowledge and skills acquired at the lower level [27].

In principle, higher-order thinking skills (HOTS) constitute a cognitive process that helps students learn complex learning concepts [28]. According to Bloom's Taxonomy, HOTS are associated with the three highest levels of mental activity, namely analysis, evaluation, and creation, thus signifying their importance to help students learn, understand, and master programming algorithms. As such, learning algorithms entails students to have good HOTS to help them learn to solve programming problems involving complex logic, which is very challenging to most students [2, 7, 9, 18, 22-24], possessing highly developed thinking skills is a prerequisite to effective learning of computer programming such that they can critically examine a given problem that leads to the finding of the best solution. According to[29], humans tend to rely on specific methods to solve problems by tapping on their prior knowledge and

experience. In principle, problem-solving methods provide the means for students to apply what they have learned (such as concepts, theories, and principles) in a new context of learning, effectively helping students to develop strong critical, analytical, logical, and rational thinking skills [30] and indirectly helping them to enhance their confidence. Therein lies the imperative for teachers and lecturers to find and use suitable teaching and learning strategies that can help students develop strong HOTS, which can help them learn more efficaciously and attain better academic achievements [7].

Arguably, one of the important aspects of learning programming courses is finding a suitable programming environment in which students can practice programming in a proper learning context. Essentially, algorithms taught in programming courses are quite different from those taught in other fields, such as mathematics. Learning algorithms in programming entails students to solve programming problems using the computer and other tools, which collectively helps them to critically and logically analyze such problems. As such, it is, therefore, important for teachers and lecturers to determine and use effective teaching methods that can assist students develop strong problem-solving skills, which ultimately can help them to learn programming algorithms with better efficacy [31].

Problem-solving models

Over recent years, problem-solving skills have been recognized as the essential learning skills that students should and must have to quickly and accurately solve problems in many domains of learning, such as mathematics and physics. To date, several problem-solving models, including Polya Model, Lester Model, and Meyer Model, have been proposed and used in the teaching and learning of various courses in a wide range of disciplines. Arguably, the use and adaptation of various problem-solving models are driven by the urgent needs to innovate existing teaching and learning techniques, with the main aim to improve students' academic achievements [14]. Arguably, the proper use of each problem-solving model relies on specific learning contexts, as determined by various aspects, such as learning topics, learning contents, levels of learning difficulty, and problem-solving processes. As such, teachers and lecturers need to identify which model is deemed most appropriate to help teach a particular topic of learning by factoring in all relevant aspects. The Polya Model, which was introduced in 1945, is one of the early problem-solving models used in solving mathematical problems involving four basic phases, namely understanding problem, action planning, action execution, and review.

Understanding problem

Arguably, students often fail to solve a given problem due to their inability to understand the constituent parts of the problem [25]. Thus, as prescribed by the Polya Model, instructions given to students must be clear and unambiguous lest the students may get confused, making it easier for them to understand the underlying aspects that define a particular problem. By understanding such a problem, students can easily identify the steps required to solve it. In the context of learning programming, the full understanding of a given programming problem can help students determine the required input, process, and output needed in developing the necessary algorithms.

Action planning

The Polya Model emphasizes on finding the appropriate strategies to help solve problems. In this second phase of problem-solving, choosing the right strategies depends on the skills learned through the iteration of the problem-solving process. Ideally, students should choose strategies that they can easily understand and apply in solving problems, which include drawing, sketching, and using formulas. In learning

programming algorithms, however, students can rely only on two strategies, namely pseudo-coding and drawing flowcharts.

Action execution

The third phase involves executing the action that has been planned in the preceding phase. This stage is quite critical in that the execution of all the necessary actions has to be carried carefully and systematically to achieve the intended outcomes. Thus, this entails programming students to rely on their skills to develop proper problem-solving strategies to help them create precise algorithms to help solve programming problems.

Review

In this last phase, students need to devote some time to reviewing and reflecting on the solutions to the problems that they have solved [15]. Such a process helps improve their confidence, strengthen skills, and ensure that problems were resolved correctly. This phase also allows students to determine and understand relevant strategies that they can use to solve other problems in the future. As such, this phase is an important step in solving programming problems that entail proper algorithms, as logical errors may occur during the development of algorithms that will be difficult to detect once they have been converted into programs.

To date, many studies have shown that the Polya Model is an effective, efficient problem-solving technique that can help students to solve a range of problems, including programming problems [2, 14]. This model enables students to describe appropriate processes and plan appropriate actions that help them develop strong critical thinking skills. Also, Whilst [12] assert that this model can help students become more inquisitive, attentive, and confident in solving problems.

Over the years, new variants of the Polya Model have been developed and used for problem-solving. These include the Lester model and Meyer model that were developed to meet the different requirements of problem-solving methods. In principle, the Lester model defines a problem as a situation where an individual or a group of individuals performs a task without referring to any algorithm or procedure. Thus, students must perform a few trials or experiments to develop a solution which can be based on abstract results. Due to this limitation, the Lester model is inappropriate for learning the topic of programming algorithms, because the final phase of the model allows students to design results independently without any conditions, which is not suitable for the development of programming algorithms as the processes are subject to two techniques only: pseudo code and flow chart. Likewise, the Mayer model is also unsuitable for learning programming algorithms. Despite being the latest version of the Polya Model, the Mayer model lacks the review phase, which is an important step to help ensure algorithms developed can meet the requirements of eliminating any logical errors. Table 1 shows the steps or phases of the problem-solving process of the Polya Model, Lester Model, and Meyer Model.

Table 1: The phases of the problem-solving process of the Polya Model, Lester Model, and Meyer Model

Steps	Polya Model	Lester Model	Meyer Model
1st Step	Understanding problem.	Problem readiness. Understanding problem. Analyzing problem.	Interpret problem. Integrate problem.
2nd Step	Action planning.	Strategy planning.	Planning and monitoring.
3rd Step	Action execution.	Strategy execution.	Execute Solution.
4th Step	Review.	Solution procedure and evaluation.	Not available

As shown, there are some subtle differences between the three problem-solving models, with each having its unique strengths that can help facilitate the learning of a particular topic. Given that the Polya Model has more detailed steps, it is deemed the most appropriate problem-solving model for learning programming algorithms. Such a choice is in line with many studies that showed the Polya Model has been widely used in various fields of study, signifying its effectiveness in helping students to improve their problem-solving skills.

METHODOLOGY

Research design

This study was based on a quasi-experimental approach using the pretest-posttest non-equivalent control group design to determine the difference in students' learning performances in programming algorithms between two groups, namely a control group and a treatment group. According to [32], researchers are compelled to use such an approach in cases where simple random sampling is not possible. In reality, true-experimental approach is hardly possible as simple randomization sampling may disrupt the learning process in the classroom, leaving researchers with no other alternatives except convenient sampling by using intact classes [33]. In addition, there are factors that many researchers have to deal with, such as time, cost, and logistics, before embarking on a true-experimental study [34]. To mitigate threats to the internal validity of the study due to the inhomogeneity of the study sample, pre-tests were used as covariates in the statistical analysis. Figure 1 shows the research design used in this study.

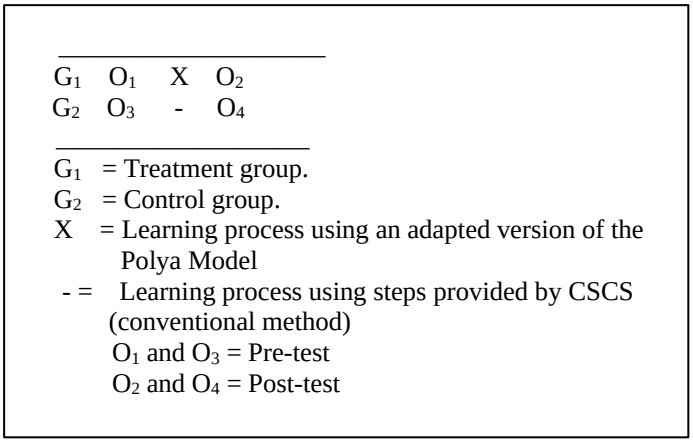


Figure 1: The quasi-experiment design of research

Sampling and population

In this study, the study sample consisted of students majoring in science in one of the matriculation colleges in Malaysia. They were purposely selected given that they were only STEM students in this college. Altogether, there were 18 running classes from which only two classes were selected based on a polling technique According to[18], such a sampling technique helps determine a study sample that is representative to a target population. In this study, 30 students were selected from the two classes, with each class consisting of 15 students.

Research Instrument

In this study, the instrument was based on an adapted version of the Polya Model, which was used in the teaching and learning process of programming algorithms. The model consists of four phases or stages, namely understanding the problem, planning the appropriate actions, executing the actions, and reviewing the outcome, which must be strictly followed as each phase helps students to learn and understand the problem-solving process systematically[34]. In this study, some minor adaptions were made in some elements of the phases to make them compatible with the requirements of CSCS [9] for the learning of programming algorithms in the matriculation program, as shown in Figure 2. Each of these proposed elements refers to the learning process of programming algorithms in general and the CSCS for the Matriculation Program in Malaysia. In addition, a total of 5 experts in Computer Science field were consulted to obtain the validity of the elements for each phase.

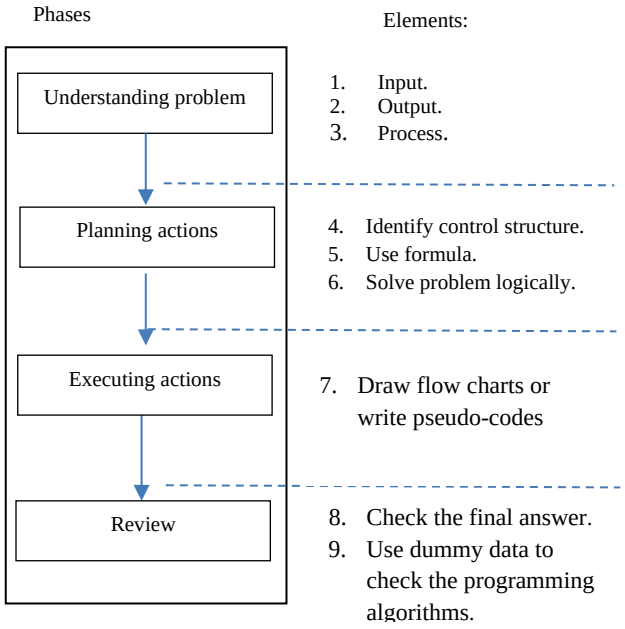


Figure 2: An adapted version of the Polya Model for the learning of programming algorithms

The pre-test and post-test questions were based on the test specification table (TST) derived from a collection of matriculation final semester examination papers developed by computer science lecturers from several universities and matriculation colleges, which had been validated by three expects. A pilot study was conducted to determine the validity and reliability of such items and to assess respondents' ability to understand and answers the test questions. A set of pre-test questions was distributed to the respondents before the learning treatment, while a set of post-test questions was distributed after the completion of the learning treatment. Their

responses to the pre-test and post-test items were analyzed, revealing Cronbach's Alpha values of 0.71 and 0.76, respectively, signifying their reliability was acceptable.

Each set of the test questions consisted of six subjective questions with three levels of difficulty (namely, low, moderate, and high levels) based on the Bloom's taxonomy, as shown in Table 2. To facilitate the process of data analysis, their responses were dichotomized to indicate a right or a wrong answer. Also, the scoring format was divided into three ranges, namely low, moderate, and high.

Table 2: The control structure of pre-test and post-test questions by the levels of Bloom Taxonomy and the types of solution

No	Control Structure	Question	Level of Bloom Taxonomy	Type of solution
1	Sequential	1	Easy	Input/Output/Process (IPO)
	Selection	2	Easy	IPO
2	Repetition	3	Intermediate	Flow chart
	Selection	4	Intermediate	Pseudo Code
3	Repetition	5	Difficult	Flow chart
	Selection & Repetition	6	Difficult	Pseudo Code

Research procedure and data collection

To meet the requirements of data, students' test results were collected in phases. As for the teaching and learning process, the treatment group learnt algorithm development using adaption of Polya model as shown in Figure 2, whilst control group were based on the conventional method using guidelines prescribed in the CSCS until second steps only without reviewing phase and these sessions were carried out in a two-hour tutorial session spanning seven (7) consecutive weeks by the same lecturers. The learning treatments were conducted at the beginning of a new semester based on the timetable set by the management of the college. Figure 3 shows the flow of the data collection procedure.

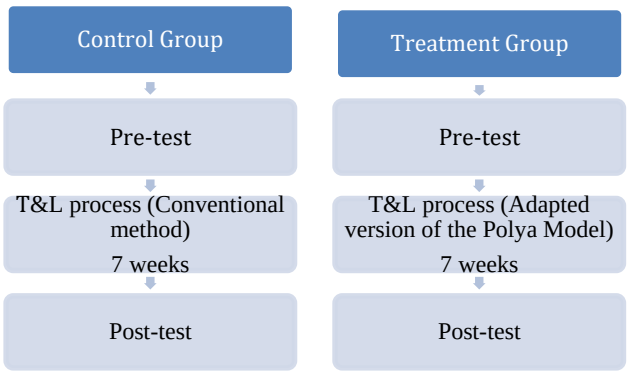


Figure 3: Data collection procedure

Data analysis

Both descriptive analysis and parametric analysis were carried out to answer the research hypotheses. Table 3 shows the types of data analysis performed to address the research questions.

-

Table 3: Data analysis for each research question

No	Research Question	Instrument	Sources for data analysis
1.	Are the elements identified and defined in every phase of the adapted Polya Model suitable for the teaching and learning of programming algorithms?	Post-test	Students' post-test answer sheets. Notations made on the answers sheets.
2.	Is there a significant difference in students' learning performance of programming algorithms between students who learned using the adapted Polya Model and those who learned using the conventional method?	Post-test	Students' marks. Notations made on the answers sheets.

RESULTS

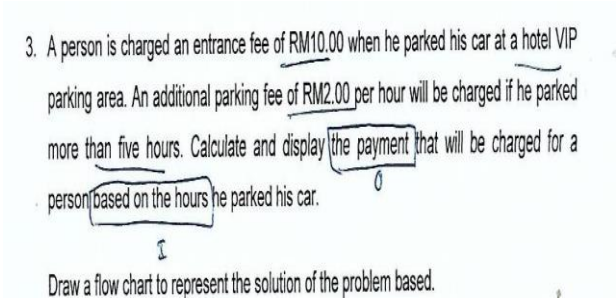
This first objective of the research is to identify and define the elements in each phase of the Polya Model that are suitable for learning programming algorithm. In the original Polya Model, there are four phases that need to be implemented, namely understanding the problem, planning the actions, implementing the actions and reviewing, some of which are deemed not suitable for the learning of programming algorithms. Thus, the researcher made some amendments to the model by adding new elements and detailed descriptions in each phase as shown in Figure 2. To ensure the elements in each phase of the adapted model would be appropriate for the learning of programming algorithm, post-test questions of the intermediate level of difficulty were chosen for the ensuing analysis. Prior to treatments, students in the treatment group were explained about the details of every phase of the adapted Polya Model that they had to follow, the solutions of which had to be demonstrated on the answer sheets. On the other hand, students in the control group learned the problem-solving principles based on the conventional method. By examining the answer sheets, it was revealed that more than half (50%) of the students in the treatment group were able to perform all the phases of problem-solving as required. By contrast, only a handful of the students in the control group, constituting about 20%, were able to do so. Table 4 summarizes the percentages of the number of students who were able to perform all phases of problem-solving.

Table 4: The percentages of the number of students who were able to perform all phases of problem-solving

Phase	Treatment Group		Control Group	
	No. of students	Percentage	No. of students	Percentage
Understanding problem	10	66.7%	5	33.3%
Planning actions	9	60.0%	3	20%
Executing actions	9	53.3%	2	13.3%
Review	9	60.0%	3	20%

The examination of the notations made on the answer sheets by the students in the treatment group, a sample of which is shown in Figure 4a, indicated that about half

(50%) of them were able to correctly identify the elements of the questions that were related to the problem-solving steps. Specifically, the examination revealed that most of the students managed to identify the required inputs and outputs, but not the processes, based on the notations that they made. Such findings, however, were not replicated by the control group, as a majority of the students were unable to identify the elements of the questions that were related to the problem-solving steps needed in developing algorithms. Figure 4b shows a sample of the notations of elements of the questions relating to the problem-solving steps made by students in the control group. These findings suggest that the first phase of problem-solving of the adapted Polya Model is appropriate for the learning of programming algorithms.


3. A person is charged an entrance fee of RM10.00 when he parked his car at a hotel VIP parking area. An additional parking fee of RM2.00 per hour will be charged if he parked more than five hours. Calculate and display the payment that will be charged for a person based on the hours he parked his car.

Draw a flow chart to represent the solution of the problem based.

Figure 4a: An example of notations made on an answer sheet by a student in the treatment group

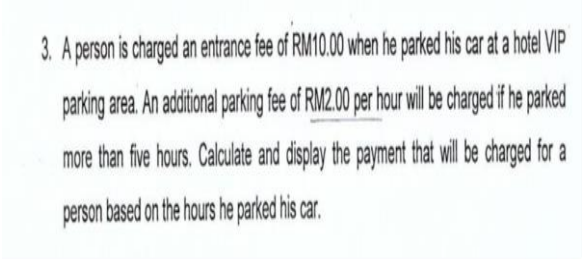
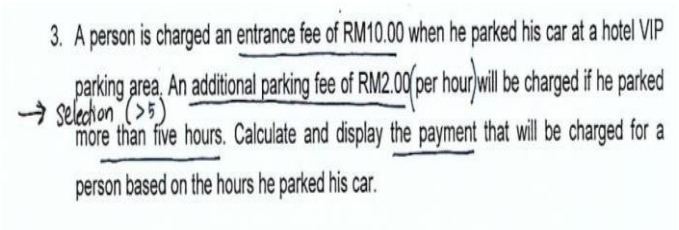

3. A person is charged an entrance fee of RM10.00 when he parked his car at a hotel VIP parking area. An additional parking fee of RM2.00 per hour will be charged if he parked more than five hours. Calculate and display the payment that will be charged for a person based on the hours he parked his car.

Figure 4b: An example of notations made on an answer sheet by a student in the control group

Next in the second phase of action planning, the researcher has identified the necessary elements which is to determine the control structure used in developing the programming algorithm. Over here, students need to identify the control structure involved as well as determine the formula or formulas that need to be used to complete the process to obtain the correct output result. Figure 5 shows the implementation of the elements in the second phase on the post-test answer sheet of the treatment group. However, students in control group did not specify any control structure that need to be applied in the solution.


3. A person is charged an entrance fee of RM10.00 when he parked his car at a hotel VIP parking area. An additional parking fee of RM2.00 per hour will be charged if he parked more than five hours. Calculate and display the payment that will be charged for a person based on the hours he parked his car.

→ Selection (>5)

Figure 5: An example of notations (identifying selection control structure) made on an answer sheet by a student in the treatment group.

The third phase of problem-solving of the adapted model is to perform the necessary actions to develop the required algorithms in the form of either pseudo-codes or flow charts based on a given question. Figure 6a and Figure 6b show two examples of flow charts drawn by a student in the treatment group and a student in the control group on their answer sheets, respectively.

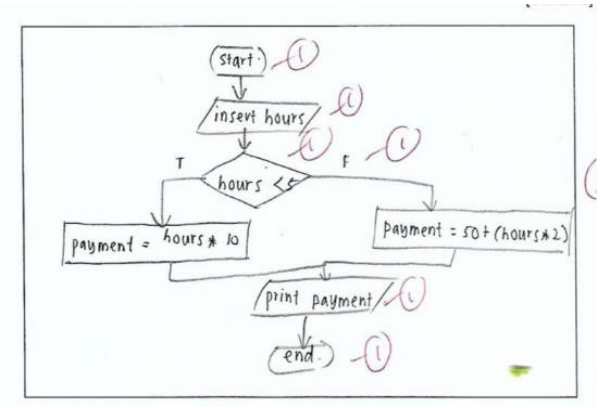


Figure 6a: An example of a flow chart drawn on the answer sheet by a student in the treatment group

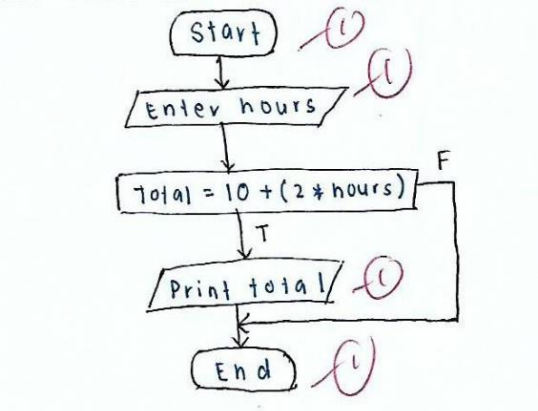


Figure 6b: An example of a flow chart drawn on the answer sheet by a student in the control group.

As clearly shown in the above figures, the student in the treatment group demonstrated a higher level of skill in developing programming algorithms compared to his counterpart in the control group. In particular, he was able to identify the correct formula and process to develop the required algorithms. Examining the answer sheets of all students in the treatment group revealed that more than 80% of them were able to develop complete algorithms correctly. By contrast, about half (50%) of the students in the control group were able to do so, which is best exemplified by Figure 6b, showing an incomplete flow chart made by one of the students in this group. Evidently, most students in the control group could only correctly identify the elements of the first phase of problem-solving, namely inputs, processes, and outputs, but they failed to identify the control structure, operations, and formulas involved in the problem-solving process, rendering them unable to plan and develop complete algorithms. This particular finding reinforces the importance in determining the proper elements in each phase of problem-solving to help students develop accurate, efficient programming algorithms.

Scrutinizing the answers sheets of students in the control group showed that most of them did not recheck their answers after they had developed the algorithms, as there were no signs or notations made on their answer sheets to suggest some

corrections were made as deemed necessary. This was in stark contrast with students in the treatment group in which 60% did recheck their works by performing one of the two reviewing methods as follows: by using dummy data in the chosen formula or process or by repeating the previous phase of problem-solving several times to obtain the correct answers (algorithms). This shows that reviewing the answer from the first step is important to get the final result correct. The proposed adaptation model in Figure 2 did not emphasize or point out that the review process must done either from the first step or any other steps. However, based on the notations that done by students from treatment group shows that the review process must either start from first step or any step which is essential to get a correct answer. Accordingly, a slight modification was made to revise the Polya Model by incorporating the above findings in its fourth phase (review phase) of the problem-solving process as shown in Figure 7. Overall, the above findings helped the researchers to validate the elements proposed for the adapted Polya Model, which were observed to be effective for the learning of the development of programming algorithms.

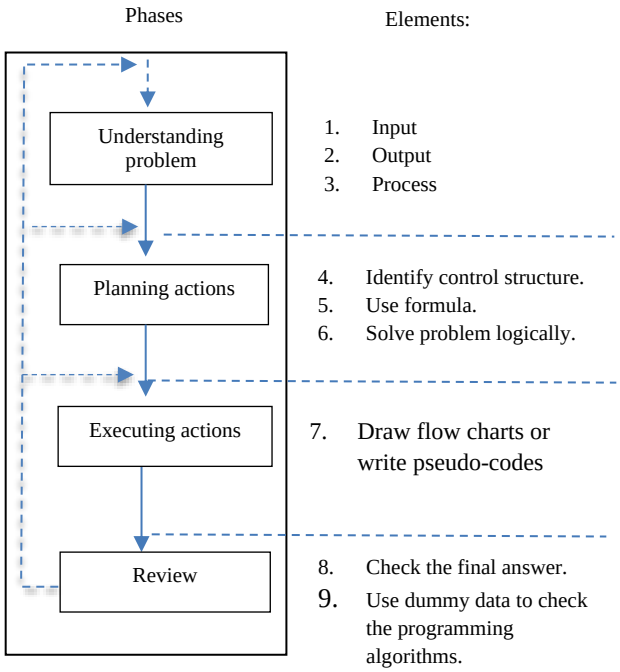


Figure 7: The revised Polya Model after reviewing students' answer sheets.

To address the second research objective, the researchers performed a series of independent-*t* tests to determine if there were significant differences in students' learning performances of programming algorithms between the two groups. An independent t-test performed on students' pre-test measurements showed that there no significant difference in their learning performances before the learning treatments, $t(28) = .621$, $p > 0.05$, signifying that both groups were equivalent in terms of their knowledge of programming algorithms. After the completion of the learning treatments, both groups were post-tested with the same research instrument. Table 5 summarizes the descriptive statistics of students' post-test measurements of learning performances in programming algorithms.

Table 5: Students' learning performances after the learning treatments

Learning method	N	Mean	Std. deviation
Conventional	15	27.6	6.057
Based on the adapted Polya Model	15	39.60	4.014

An ensuing independent *t*-test performed on the students' post-test measurements showed there was a significant difference in the mean scores of learning performance in programming algorithms between the treatment group and the control group, t (28) = -6.39, $p < 0.001$, thus providing the evidence to reject the null hypothesis. In other words, students in the treatment group outperformed those in the control group. Table 6 summarizes the results of the independent *t*-test performed on students' post-test measurements.

Table 6: The results of the independent *t*-test of students' learning performances after learning treatments.

Measure	t	df	Sig
Learning performance in programming algorithms.	-6.39	28	0.0001

Detailed descriptive analysis was also performed to highlight the differences in students' mean scores of learning performance based on the level of difficulty. As discussed earlier, the set of questions used for pre-testing and post-testing consisted of six questions with three levels of difficulty based on Bloom's Taxonomy, namely low, intermediate, and high, with each being represented by two questions. Table 7 summarizes the mean percentage points of students' learning performances in the three categories of questions after the learning treatments.

Table 7: Mean percentage points of students' learning performances based on three levels of difficulty.

Group	Level of difficulty		
	Low	Intermediate	High
Treatment	100	72.9	81.39
Control	89.17	56.67	42.78

As shown, students in both groups scored high mean percentage points (100 vs. 9.17) for the first two questions that were based on the low level of difficulty, the difference of which was 10.83%, which was considered quite small. For the third and fourth questions of intermediate level of difficulty, their mean percentage points (72.9 vs. 56.67) dropped quite significantly, resulting in a difference of 16.3%. Such a drop in performance indicated that students, especially in the control, had some difficulties to solve the questions at this level of difficulty. For the remaining last two questions of

high level of difficulty, the mean percentage points of learning performance of students in the treatment group and in the control group dropped to 81.39 and 42.78, respectively. Clearly, the drop in learning performance of the former was quite small compared to that of the latter, which at 38.61% was massive. This finding indicated that students in the control group were overwhelmed by the difficult questions, making them unable to solve such questions correctly. Overall, the above findings suggest that the use of the adapted Polya Model can help programming students gain strong HOTS, with which they can solve a range of problems in programming algorithms, especially those of high level of difficulty.

DISCUSSION

In principle, the Polya Model involves four phases of problem-solving, namely understanding the problem, planning actions, executing actions, and reviewing. In this study, the researchers used the same terminology for each phase of the problem-solving process but the elements in each phase were adapted to the learning of the topic of algorithms to ensure students can learn to solve programming problems more easily and systematically. The analysis of students' notations on the post-test answer sheets in the treatment group showed they were able to identify such elements correctly, indicating that the elements of first phase of the problem-solving process helped them to understand the problems posed in the questions more clearly. Evidently, such an understanding of problems in this phase helped them to confidently proceed to the next phase of the problem-solving process. By contrast, most students, except a handful, in the control group were not able to clearly understand the same problem, given the lack of evidence in the form of notations made on their answer sheets. This finding suggests that the conventional learning method does not provide a clear guideline to help students discern the appropriate requirements of how to solve questions (especially those of high level of difficulty) at the early stage of the problem-solving process. Clearly, a lack of such discernment will complicate students' attempts to carry out the next phase of the problem-solving process.

In the second phase of the Polya Model, namely the action planning phase, students were required to determine the control structure involved in the construction of algorithms, which could be a single control structure, a sequence of control structures or a repetition of control structures. This process of determining the appropriate control structure was proposed by Yew [36], who stressed its importance in ensuring the final algorithms developed would free of errors. The analysis of post-test answer sheets of students in the treatment group showed that all of them managed identify the control structures involved correctly for all questions. On the other hand, the same analysis showed only a handful in the control group were able to do so. More revealingly, it was observed that there were some students in this group who did use any form of control structure in developing their algorithms, signifying their inability to decompose the given problems. Clearly, such a failure in this phase would be detrimental to their efforts to successfully carry out the ensuing phases of the problem-solving process.

The ensuring phase, namely the execution phase, of this adapted model entailed students to develop the required algorithms by identifying the relevant types of algorithms which could be developed in either pseudo-codes or flowcharts. To perform these tasks, the student had to rely on the elements that they had identified earlier in the first and second phases. Surely, the development of complete, correct algorithms depended on their abilities to identify such elements to meet the requirements of the given questions. The analysis of students' post-test answers of the treatment group showed that they were able to develop correct, complete algorithms. Arguably, this stemmed from students having correctly identified the appropriate inputs, outputs, and processes in the first phase and the correct control structure in the second phase. By contrast, more than half of students in the control group were unable to develop such

algorithms. It could be reasonably argued that they, unlike their counterparts in the treatment group, had not been able to correctly determine the required inputs, outputs, processes, and control structure in the preceding phases.

The fourth and final phase of the adapted Polya Model entailed students to review of the algorithms that had been developed. The analysis of students' post-test answers of the treatment group showed they applied the two reviewing methods taught in their classes. The first method involved using dummy data as substitute values to test whether the development process of algorithms chosen was correct. On the other hand, the second method involved repeating the steps taken in the previous phase to finally produce complete, precise algorithms. These findings demonstrated that students did review their answers (algorithms), helping them to identify the inaccuracies of or errors in their algorithms. Such reviewing steps were, however, not performed by their counterparts in the control group, as evidenced by a lack of evidence in terms of notations scribbled on the latter's answer sheets. Arguably, this particular finding might be attributed to a lack of emphasis by the conventional learning method on the needs to review algorithms that had been developed. Moreover, many strategies, models, or techniques that commonly used in learning programming does not emphasize in reviewing the algorithm [2, 3, 9-12] Mostly the focus is on programming coding and the reviewing process is done after the code is developed which is debugging process. The disadvantage of this strategy is that the students will be more focusing on syntax errors instead of logic errors. Furthermore, by reviewing the algorithm before the coding the phase, chances of logic error occurring in coding can be reduced.

Overall, the above findings showed the use of the detailed elements of the Polya Model in the learning of algorithms helped guide students in the treatment group to solve programming problems more systematically and enhance their problem-solving skill. By using such elements of the problem-solving phases of the adapted model, they were able to apply the principles and concepts of algorithmic programming that they had learned to correctly determine the correct solutions in developing precise algorithms. As such, these findings underscore the imperative of using relevant problem-solving techniques or models in the teaching and learning of computer programming, which have been widely used in previous studies [8, 16, 22, 23, 25, 34].

Another important revelation of this study concerns the differences in students' learning performances between the treatment and control groups, with the former outperforming the latter. Such differences were not significant for questions of low and intermediate levels of difficulty based on Bloom Taxonomy. But for questions of high level of difficulty, the differences were massive. Arguably, from the cognitive perspectives, solving problems based on the first two levels of difficulty only necessitated low-level thinking skills. By contrast, higher-order thinking skills were required to help students solve problems of high level of difficulty, with which they could analyze such problems more critically.

According to Persaud [27], the highest cognitive levels of Bloom Taxonomy are evaluation and creation. Arguably, students in the treatment group might have reached such levels, as they had successfully created appropriate pseudo-codes and flowcharts by incorporating several control structures in the solution to problem given in the final question of the post-test. Effectively, this showed that their strict adherence to the problem-solving steps of the Polya Model helped them to develop or create complete working algorithms. By contrast, most students in the control group did not answer this question, indicating that they lacked the thinking skills required to help them identify and use the elements of logic in generating the proper solution, which they should have done in the previous phase. The above findings are consistent with

previous findings that show the use of appropriate problem-solving models, including the Polya Model, in the teaching and learning of computer programming can enhance students' understanding and improve their learning performances [2, 18, 31]

As demonstrated in this study, the use of the Polya Model can assist students to think at the highest level of thinking, which can certainly help them to solve complex, difficult problems [7]. Likewise, the use of such of a model can help students to improve their problem-solving skills, which had been demonstrated in this study that showed students in the treatment group were able to perform the abstraction process in solving the problems. Additionally, as revealed in their answer's sheets, they seemed to possess a firm understanding of the concept of programming at the early stage of solving the programming problems before they started the coding process. In this regard, according to Stefania et al. [16], a lack of exposure to computing features will make it difficult for students to understand the underlying concepts of programming. Thus, it can be reasoned that the use of the Polya Model can help students learn programming languages more efficaciously by focusing on basic programming concepts, such as the control structure, before they start learning more difficult or abstract programming concepts, such as functions, arrays, and pointers

Finally, the findings showed that the use of Polya Model helped expose students to the proper learning cycle that trained them to perform all the steps of the problem-solving phases by first completing the steps of the first phase before proceeding the subsequent phases. Effectively, by strictly following the learning cycle, students can understand the basic underlying concepts and acquired the necessary skills, as emphasized by [24]. Collectively, the findings of this research suggest that the use of such a model is effective in enhancing students' problem-solving skills, which tends to concur with the recommendations made by other researchers [10, 15, 17]. The imperative to use the Polya Model has become more pressing of late, given that the learning of computer programming has been found to be extremely challenging [31, 33]due to a lack of problem-solving skills among programming students [10, 14]The findings of this study also revealed by tapping on HOTS students make connections between new knowledge and previous ones, thus allowing them to construct meaningful learning. Thus, we can conclude that the use of Polya Model adaptation helps students master programming skills and knowledge more quickly and meaningfully.

CONCLUSION

Obviously, effective teaching and learning methods are essential to helping students, especially novice students, to learn challenging, difficult courses at the collegiate level, such as computer programming. As demonstrated in this study, the use of an adapted version of the Polya Model as an innovative problem-solving method in the teaching and learning of the development of programming algorithms can help students learn more systematically and meaningfully. Finding shows by using Polya model as a teaching and learning method in algorithm development in the matriculation center has helped the students' enhance their skill in programming. In other words, the elements of the problem-solving phases of the model were made more explicit and detailed to guide students to solve programming problems more effectively, which they demonstrated by their abilities to create accurate algorithms, the process of which helped them develop strong HOTS. Thus, the findings of this study reinforce the importance in tailoring any problem-solving methods, such as the Polya Model, to meet the specific needs and requirements of a particular course, which in the final analysis can help improve students' learning performances and thinking skills.

ACKNOWLEDGMENT

Authors wish to thank Sultan Idris Education University, Malaysian Ministry of Education and Malaysian Matriculation Center for their support in this study.

REFERENCES

1. Omar, R., et al., *Comparison of visual aids for improving reading performance in children with dyslexia.* Medical hypothesis, discovery & innovation in optometry, 2021. **2**(2): p. 85-93.DOI: https://doi.org/10.51329/mehdioptometry130.
2. Kanika, S. Chakraverty, and P. Chakraborty, *Tools and techniques for teaching computer programming: A review.* Journal of Educational Technology Systems, 2020. **49**(2): p. 170-198.DOI: https://doi.org/10.1177/0047239520926971.
3. Ubaidullah, N.H., et al., *Improving Novice Studentsâ€™ Computational Thinking Skills by Problem-Solving and Metacognitive Techniques.* International Journal of Learning, Teaching and Educational Research, 2021. **20**(6).DOI: https://doi.org/10.26803/ijlter.20.6.5.
4. Kaptiningrum, P. and Z. Mubarok, *Efektifitas program matrikulasi bahasa untuk meningkatkan kemampuan speaking mahasiswa STAIBN tegal.* SHAHIH: Journal of Islamicate Multidisciplinary, 2016. **1**(2): p. 149-165.DOI: https://doi.org/10.22515/shahih.v1i2.460.
5. Goldstein, P.A., C. Storey-Johnson, and S. Beck, *Facilitating the initiation of the physician's professional identity: Cornell's urban semester program.* Perspectives on medical education, 2014. **3**(6): p. 492-499.DOI: https://doi.org/10.1007/s40037-014-0151-y.
6. Kaptiningrum, P. and Z. Mubarok, *The effectiveness of the language matriculation program to improve the speaking skills of STAIBN Tegal students.* SHAHIH: Journal of Islamicate Multidisciplinary, 2016. **1**(2): p. 149-165.DOI: https://doi.org/10.22515/shahih.v1i2.460.
7. Chinedu, C.C., O.S. Olabiyi, and Y. Kamin, *Strategies for improving higher order thinking skills in teaching and learning of design and technology education, 7(2), 35-43.* 2015.
8. Talib, N., et al., *Integrating technological pedagogical and content knowledge in computer programming courses: Issues and challenges.* Journal of Advanced Research Design, 2016. **27**(1): p. 1-15.
9. Fluck, A., et al., *Arguing for computer science in the school curriculum.* Journal of educational technology & society, 2016. **19**(3): p. 38-46.
10. Sun, D., et al., *Three contrasting pairs' collaborative programming processes in China's secondary education.* Journal of Educational Computing Research, 2021. **59**(4): p. 740-762.DOI: https://doi.org/10.1177/0735633120973430.
11. Vasilopoulos, I.V. and P. Van Schaik, *Koios: Design, development, and evaluation of an educational visual tool for Greek novice programmers.* Journal of Educational Computing Research, 2019. **57**(5): p. 1227-1259.DOI: https://doi.org/10.1177/0735633118781776.
12. Malik, S.I. and J. Coldwell-Neilson, *A model for teaching an introductory programming course using ADRI.* Education and Information Technologies, 2017. **22**(3): p. 1089-1120.DOI: https://doi.org/10.1007/s10639-016-9474-0.
13. Mustika, I.K.A. and P.N. Riastini, *Pengaruh model Polya terhadap kemampuan pemecahan masalah matematika siswa kelas V SD.* International Journal of community service learning, 2017. **1**(1): p. 31-38.DOI: https://doi.org/10.23887/ijcsl.v1i1.11897.
14. Olaniyan, A.O., E.O. Omosewo, and L.I. Nwankwo, *Effect of Polya Problem-Solving Model on Senior Secondary School Students' Performance in Current Electricity.* European Journal of Science and Mathematics Education, 2015. **3**(1): p. 97-104.
15. Prahani, B.K., et al., *Effectiveness of physics learning material through guided inquiry model to improve student's problem solving skills based on multiple representation.* International journal of education and research, 2016. **4**(12): p. 231-244.
16. Siozou, S., N. Tselios, and V. Komis, *Effect of algorithms' multiple representations in the context of programming education.* Interactive Technology and Smart Education, 5(4), 230-243. , 2008.DOI: https://doi.org/10.1108/17415650810930910.
17. Wang, X.-M. and G.-J. Hwang, *A problem posing-based practicing strategy for facilitating students' computer programming skills in the team-based learning mode.* Educational Technology Research and Development, 2017. **65**(6): p. 1655-1671.DOI: https://doi.org/10.1007/s11423-017-9551-0.
18. Polya, G., *How to solve it: A new aspect of mathematical method*. 2004: Princeton university press.
19. Rudtin, N.A., *Penerapan langkah Polya dalam model problem based instruction untuk meningkatkan kemampuan siswa menyelesaikan soal cerita persegi panjang.* Jurnal Elektronik Pendidikan Matematika Tadulako, 2013. **1**(1): p. 18-33.
20. Kulsum, N.U. and K. Kristayulita, *Student Problem Solving Analysis by Step John Dewey Reviewed from Learning Style.* IJECA (International Journal of Education and Curriculum Application), 2019. **2**(2): p. 20-30.DOI: https://doi.org/10.31764/ijeca.v2i2.2102.

21. Alkhateeb, M.A. and A.M. Al-Duwairi, *The Effect of Using Mobile Applications (GeoGebra and Sketchpad) on the Students' Achievement.* International Electronic Journal of Mathematics Education, 2019. **14**(3): p. 523-533.DOI: https://doi.org/10.29333/iejme/5754.
22. Kaviza, M., *Motivasi intrinsik dan kemahiran berfikir kritis dalam pembelajaran sejarah berasaskan analisis sumber-sumber teks: Satu kajian faktorial.* Jurnal Pendidikan Bitara UPSI, 2020. **13**(1): p. 17-26.
23. Moi, C.S. *Application of polya problem solving model in computer programming topic*.
24. Yamashita, K., et al., *Classroom practice for understanding pointers using learning support system for visualizing memory image and target domain world.* Research and Practice in Technology Enhanced Learning, 2017. **12**(1): p. 1-16.DOI: https://doi.org/10.1186/s41039-017-0058-4.
25. Ivanović, M., et al., *Technology enhanced learning in programming courses–international perspective.* Education and Information Technologies, 2017. **22**(6): p. 2981-3003.DOI: https://doi.org/10.1007/s10639-016-9565-y.
26. Athreya, B., H and M. Chrystalla, *What Is Thinking? In Thinking Skill for Digital Generation: The Development of Thinking and Learning in the Age of Information, pg 25-35, Springer, Switzerland.* 2017.DOI: https://doi.org/10.1007/978-3-319-12364-6_3.
27. Tarman, B. and B. Kuran, *Examination of the cognitive level of questions in social studies textbooks and the views of teachers based on bloom taxonomy.* Educational Sciences: Theory & Practice, 2015. **15**(1).
28. Tajudin, N.a.M. and M. Chinnappan, *The Link between Higher Order Thinking Skills, Representation and Concepts in Enhancing TIMSS Tasks.* International Journal of Instruction, 2016. **9**(2): p. 199-214.DOI: https://doi.org/10.12973/iji.2016.9214a.
29. Ali, R., A. Akhter, and A. Khan, *Effect of using problem solving method in teaching mathematics on the achievement of mathematics students.* Asian Social Science, 2010. **6**(2): p. 67.DOI: https://doi.org/10.5539/ass.v6n2p67.
30. Ismail, S. and A. Atan, *Application of problem solving approach in the teaching of technical and vocational subjects in the Faculty of Education UTM.* Journal of Educational Psychology and Counseling, 2011. **2**(1): p. 113-144.
31. Grasas, A. and H. Ramalhinho, *Teaching distribution planning: a problem-based learning approach.* The International Journal of Logistics Management, 2016.DOI: https://doi.org/10.1108/IJLM-05-2014-0075.
32. Clarke, E. and J. Visser, *Pragmatic research methodology in education: possibilities and pitfalls.* International Journal of Research & Method in Education, 2019. **42**(5): p. 455-469.DOI: https://doi.org/10.1080/1743727X.2018.1524866.
33. Glover, C.M., et al., *The Health Equity Through Aging Research And Discussion (HEARD) Study: A Proposed Two-Phase Sequential Mixed-Methods Research Design To Understand Barriers And Facilitators Of Brain Donation Among Diverse Older Adults: Brain donation decision making among diverse older adults.* Experimental aging research, 2020. **46**(4): p. 311-322.DOI: https://doi.org/10.1080/0361073X.2020.1747266.
34. Polya, G., *How to solve it second edition,5-18.* 1973, Princeton University Press New Jersey.