Received: 18 April 2020, Accepted: 10 June 2020, Published 17 June 2020

ASSESSMENT OF USING *EZ-PROG*: AN EASY COLOR SCHEMATIC MODEL FOR PROGRAMMING PROBLEM SOLVING

Siti Nordianah Hai Hom[1*], Hasnul Hadi Ibrahim[2], Abrizah Ibrahim[3], Mudiana Mokhsin[4] & Corrienna Abdul Talib[5]

[1 2 3] Computer Science Unit, Department of Mathematics, Kolej Matrikulasi Johor, 84900 Tangkak, Johor, Malaysia,
[3] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia,
[4] Faculty of Social Sciences & Humanities, Universiti Teknologi Malaysia 81310 Johor Bahru, Johor.

Abstract

Excellent academic achievement can have a great impact on matriculation students and influence their decision-making in the next step at the university and the future. However, poor problem-solving skills in programming subjects makes it difficult to make good decisions. Therefore, this study was conducted to identify the effect of using *EZ-Prog* in programming problem-solving learning among matriculation students. Students were guided by the color scheme model during the programming problem-solving process. The study population came from the Kolej Matrikulasi Johor (KMJ) and the sample involved was the 91 students from the One-Year Science Program, Module Two. The experiments were conducted in three weeks. The pretest and posttest question set consisted of three subjective questions and classified into three levels of difficulty according to Bloom's Taxonomy. Data were analyzed using SPSS. The results showed that there was a significant difference between the pre and posttest results after using the *EZ-Prog*. This demonstrates the *EZ-Prog* technique used in learning programming problem-solving learning enhances student problem-solving skills. The implications of this study indicate that *EZ-Prog* can be used in learning and teaching algorithms, especially programming problems. Finally, the positive findings from the use of *EZ-Prog* indicate that this technique can be used as an innovative in teaching programming as it enhances problem-solving skills among students. This method is not limited to matriculation students, but also all students that learn a programming language and not limited to one programming language.

Keywords: Algorithms, Polya Model, Problem Solving Skill, Programming Problem Solving, Programming

1.0 INTRODUCTION

Programming Problem Solving is a cognitive process for problem-solving skills in computer science subjects starting by reading the questions, understanding the problem, planning to solve it, and writing the program to solve it. Mayer (1994) stated that problem-solving is a process by which a student performs cognitive activities aimed at overcoming the barriers that exist between the initial and the final stages of a problem. Gagne (1977) introduced eight stages of learning including problem-solving concepts. The Polya model is widely used in problem-solving (Olaniyan, Omosewo, & Levi, 2015; Widiana, Japa, Suarjana, & Diputra, 2018). According to Prahani, Limatahu, Yuanita, and Nur (2016), the Polya model enables students to describe processes, plan and eventually develop flexible thinking and skills processes that can be a driver in problem-solving. It is supported by Agus and Nanci (2017) that this model is able to enhance student engagement, attention and interest as well as enhance student self-confidence in problem-solving. The use of the Polya Model in solving computer science questions in the form of problem-solving can help students to gradually develop those questions following Gagne's Problem-Solving Learning Theory (1977) which emphasizes that learning should start from simple (basic) to complex and more complex.

The *EZ-Prog* emphasizes five step-problems solving for problem-solving to be well integrated with the Polya Model in improving problem-solving skills, especially in the fourth step of reviewing to help students revise their answers and diversify their solution strategies. It is a schematic solution with color schemes. It uses schematic colors according to the inputs (green), processes (red) and outputs (blue) that are built. Students should gradually be exposed to accurate and easy techniques and rules. Students are emphasized by memorizing schematic colors, algorithmic formats, applying schematic colors in writing and writing from problem analysis, design solution and implementation step. The students are given the practice of making *EZ-Prog* consistently. This will help students who cannot memorize programming writing techniques using old techniques and increase their effectiveness in learning Computer Science. Students do not only learn how to build algorithms, but how to use to write programming code. Students also can solve problems from simple to difficult level of problem statement using *EZ-Prog*.

The use of *EZ-Prog* has positive implications for students in shaping student identity and mastery of skills. This finding indicates that there is room for understanding the effectiveness of using *EZ-Prog* in solving programming problems.

2.0 METHODOLOGY

The study population came from the Kolej Matrikulasi Johor (KMJ) and the sample involved was the 91 students in the One Year Science Program of Module Two. The experiments were conducted in three weeks. The students were asked to solve three questions consisting of subjective questions and classified into three levels of difficulty according to the Bloom's Taxonomy that consists of a simple, medium and difficult level of questions, to determine clearly in the students' understanding of solving programming problems. In this study, students' ability to solve a programming problem was measured before using the *EZ-Prog* technique (pretest) and after using the *EZ-Prog* technique (posttest).

The pretest and posttest were based on the Matriculation Computer Science curriculum. The time allocated for each type of test was 30 minutes. Students were also required to show the steps of how they get their answers. The pretest and posttest responses were analyzed and compared against each other to measure changes in students' achievement.

Students complete the exercise using EZ-Prog (Refer to Figure 1). The technique starts with writing the problem statement. After that, the students determine and underlined the output (blue) and input (green) (Refer to Figure 2). Students need to identify the control structure involved (Refer to Figure 3). This technique has five steps followed by:

1. Problem Analysis

   An analysis of a problem statement to identify input, process and output.
   - input – what data is needed to get the output.
   - process - explanation of how to process the data to get the output.
   - output. – what to find or calculate.
2. Design a solution

   Creating an algorithm is a sequence of well-defined steps to solve a problem. The algorithm can be represented using
   - pseudocode - An artificial informal language used to develop algorithm
   - or flowchart - A graphical representation of the algorithm connected by using flow lines.
3. Implementation

   Implementation is the process of implementing an algorithm by writing a computer program using a programming language (for example, using Java language).

4. Testing

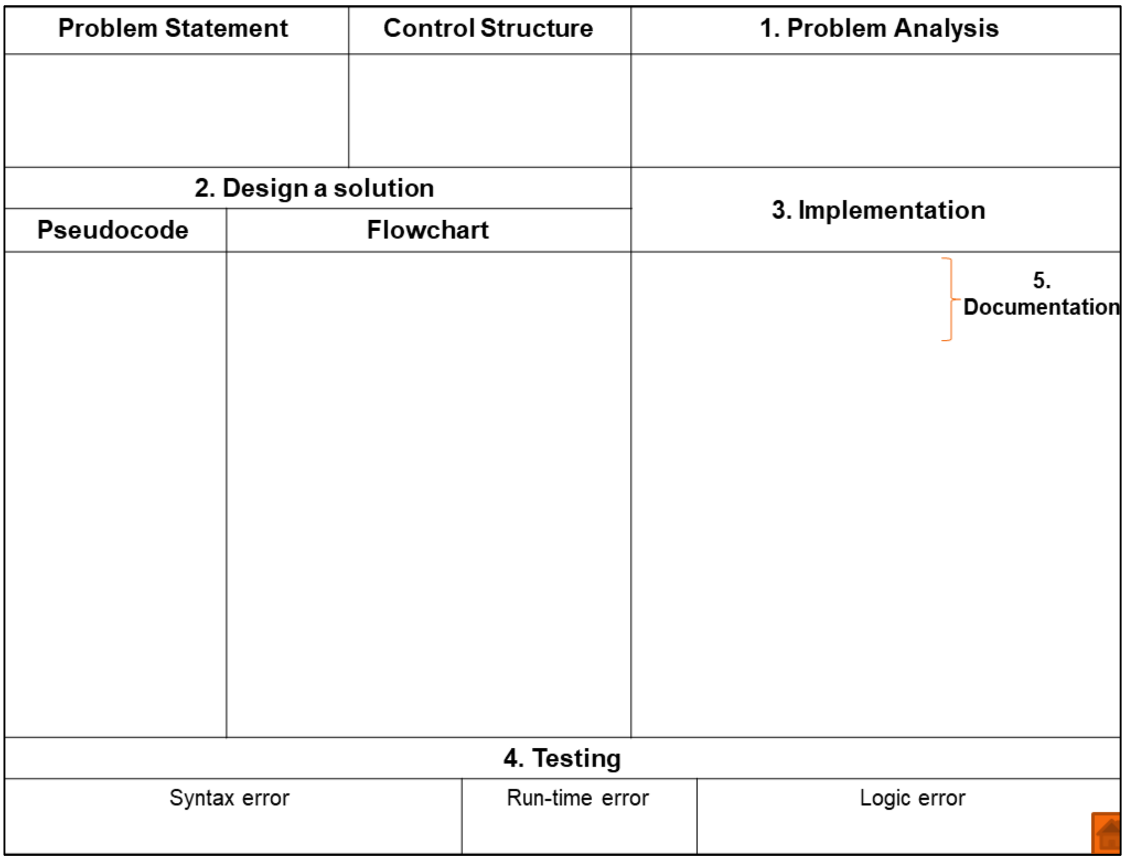
   Program testing is the process of executing a program to demonstrate its correctness (i.e. run the program). Program verification is the process of ensuring that a program meets user-requirement (i.e. test/verify it with different inputs).

   - Syntax Error - Occurs when the code violates the syntax or grammar of the programming language, occurs when the programmer fails to obey one of the grammar rules of the language.
   - Logic Error - Flaw in program design that causes an inaccurate result.
   - Run time Error - Program error or event that causes the program to stop running, occurs whenever the program instructs the computer to do something that it is either incapable or unwilling to do.

5. Documentation

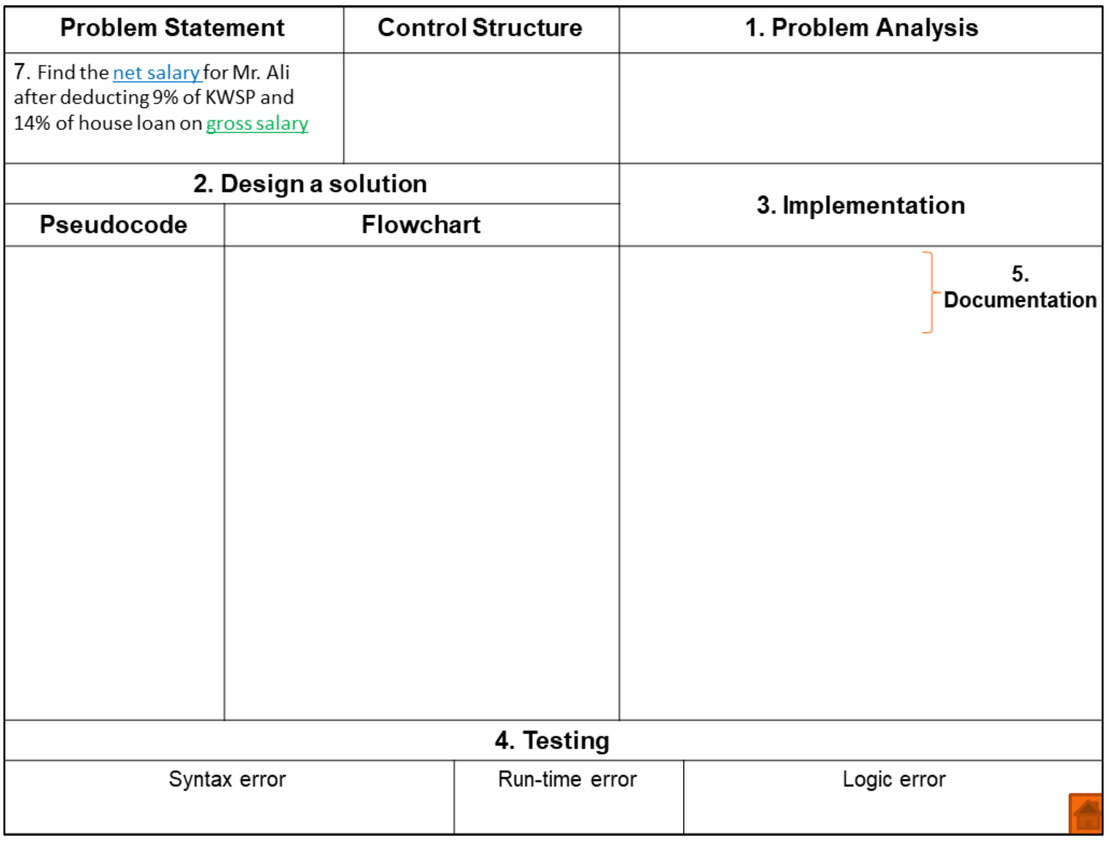
   The documentation contains a description of the program that helps other programmers in editing or maintaining the program later. Can be done in 2 ways:

   - Writing comments between your lines of codes
   - Creating a separate text file to explain the program



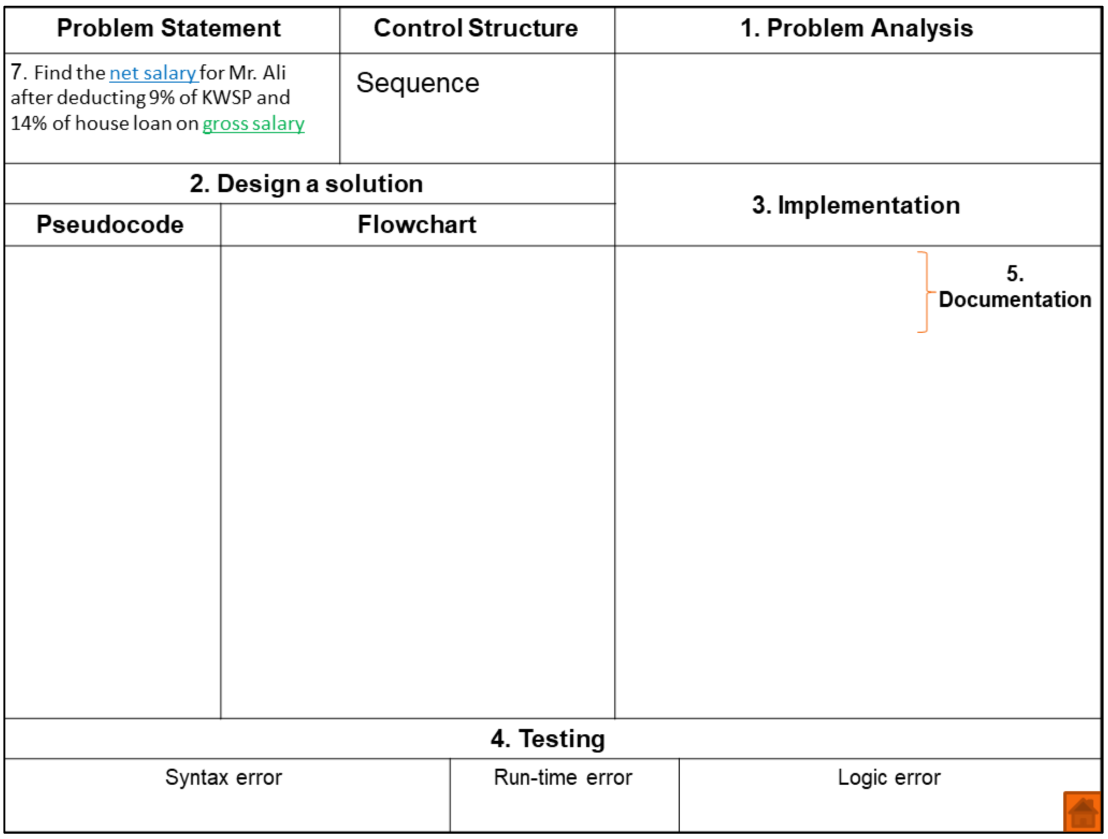
Problem Statement	Control Structure	1. Problem Analysis
2. Design a solution		**3. Implementation**
Pseudocode	**Flowchart**	
		5. Documentation
4. Testing		
Syntax error	Run-time error	Logic error

Figure 1: Template of *EZ-Prog* color schematic model for programming problem solving



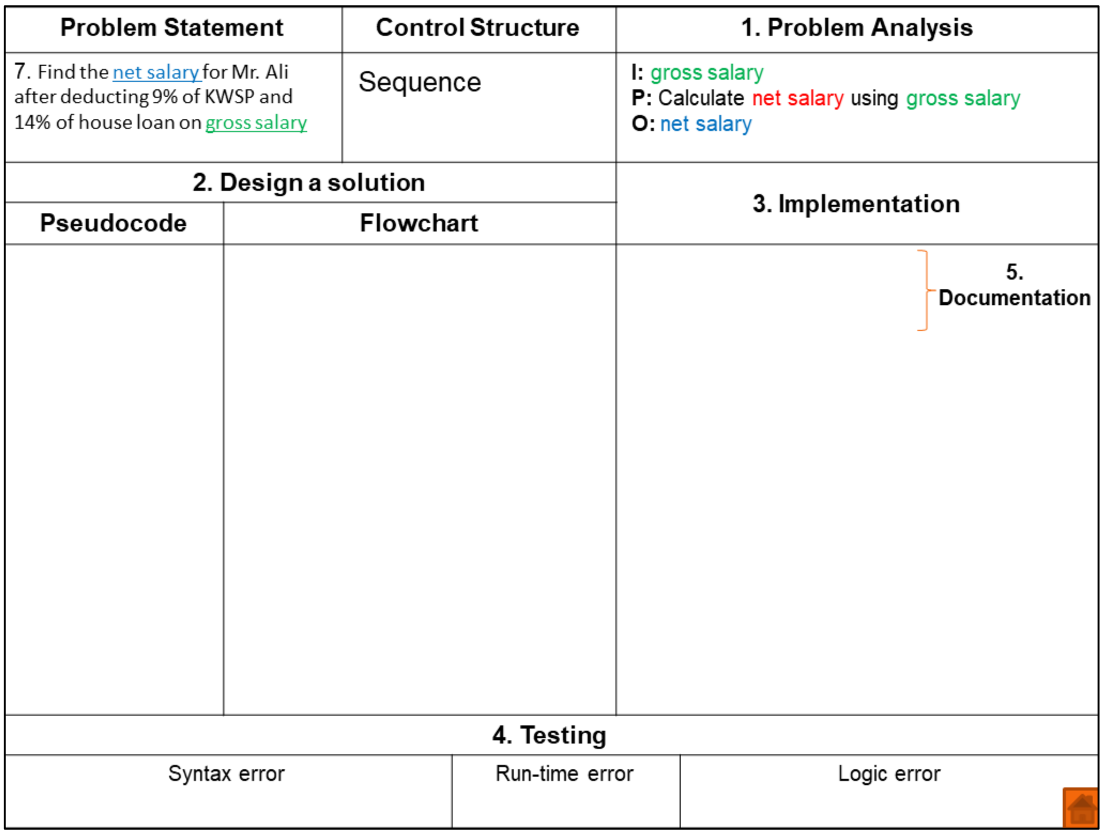
Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary		
2. Design a solution		**3. Implementation**
Pseudocode	**Flowchart**	
		5. Documentation
4. Testing		
Syntax error	Run-time error	Logic error

Figure 2: Problem statement



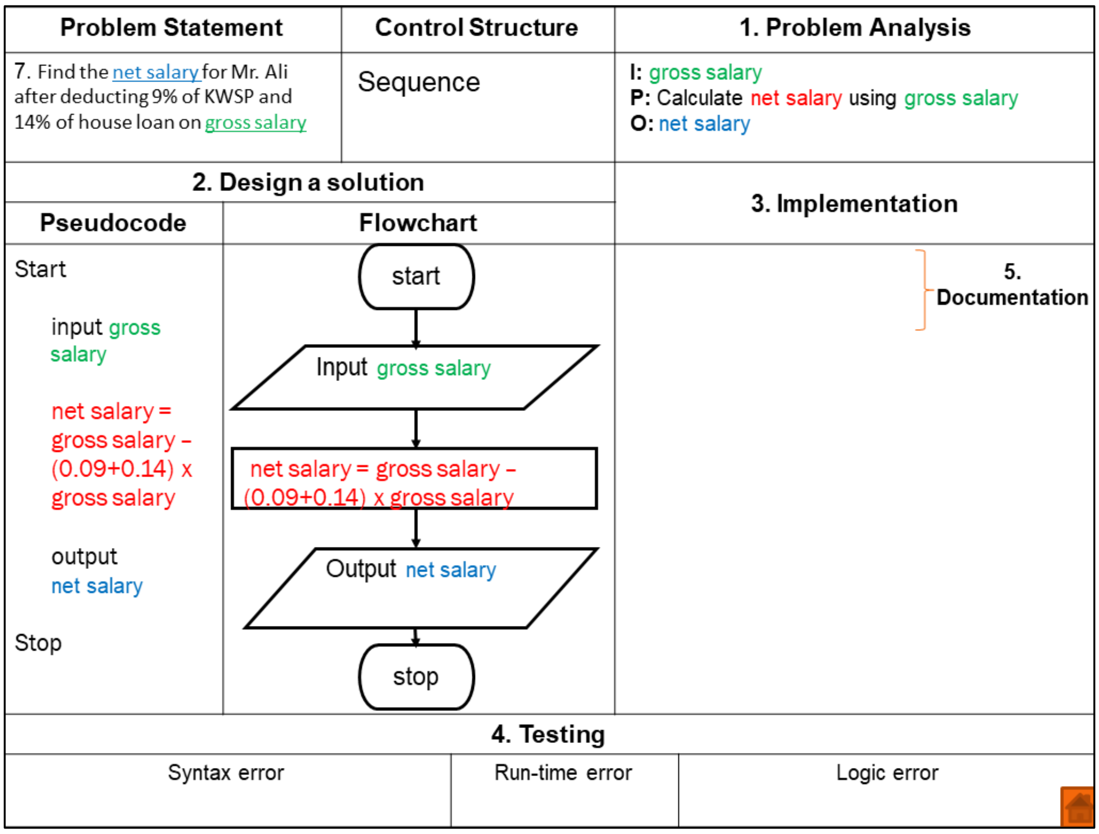
Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary	Sequence	
2. Design a solution		**3. Implementation**
Pseudocode	**Flowchart**	
		5. Documentation
4. Testing		
Syntax error	Run-time error	Logic error

Figure 3: Control Structure



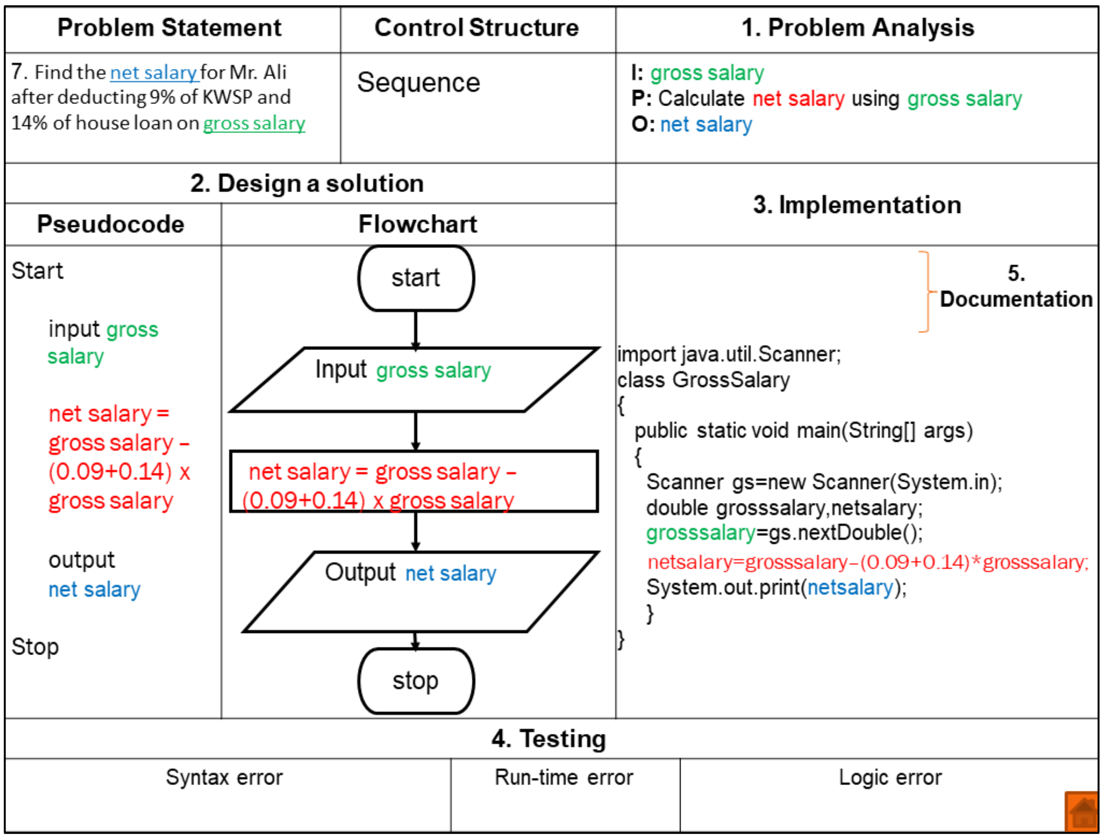
Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary	Sequence	**I:** gross salary **P:** Calculate net salary using gross salary **O:** net salary
2. Design a solution		**3. Implementation**
Pseudocode	**Flowchart**	
		5. Documentation
4. Testing		
Syntax error	Run-time error	Logic error

Figure 4: Step 1 Problem Analysis



Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary	Sequence	**I:** gross salary **P:** Calculate net salary using gross salary **O:** net salary
2. Design a solution		**3. Implementation**
Pseudocode	**Flowchart**	
Start input gross salary net salary = gross salary – (0.09+0.14) x gross salary output net salary Stop	start Input gross salary net salary = gross salary – (0.09+0.14) x gross salary Output net salary stop	**5. Documentation**
4. Testing		
Syntax error	Run-time error	Logic error

Figure 5: Step 2 Design a solution



Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary	Sequence	**I:** gross salary **P:** Calculate net salary using gross salary **O:** net salary

2. Design a solution — **Pseudocode**

```
Start
    input gross salary
    net salary = gross salary – (0.09+0.14) x gross salary
    output net salary
Stop
```

Flowchart

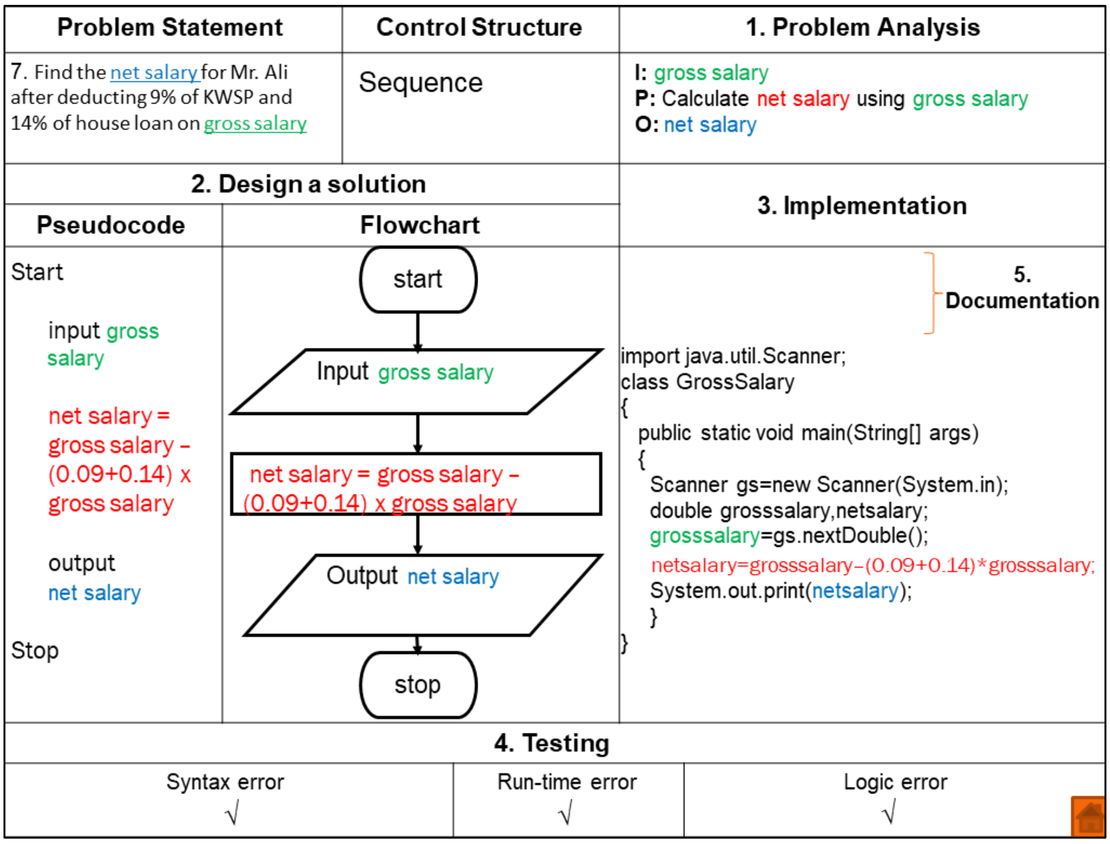
3. Implementation — **5. Documentation**

```
import java.util.Scanner;
class GrossSalary
{
   public static void main(String[] args)
   {
    Scanner gs=new Scanner(System.in);
    double grosssalary,netsalary;
    grosssalary=gs.nextDouble();
    netsalary=grosssalary-(0.09+0.14)*grosssalary;
    System.out.print(netsalary);
    }
}
```

4. Testing

Syntax error	Run-time error	Logic error

Figure 6: Step 3 Implementation



Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary	Sequence	**I:** gross salary **P:** Calculate net salary using gross salary **O:** net salary

2. Design a solution — **Pseudocode**

```
Start
    input gross salary
    net salary = gross salary – (0.09+0.14) x gross salary
    output net salary
Stop
```

Flowchart

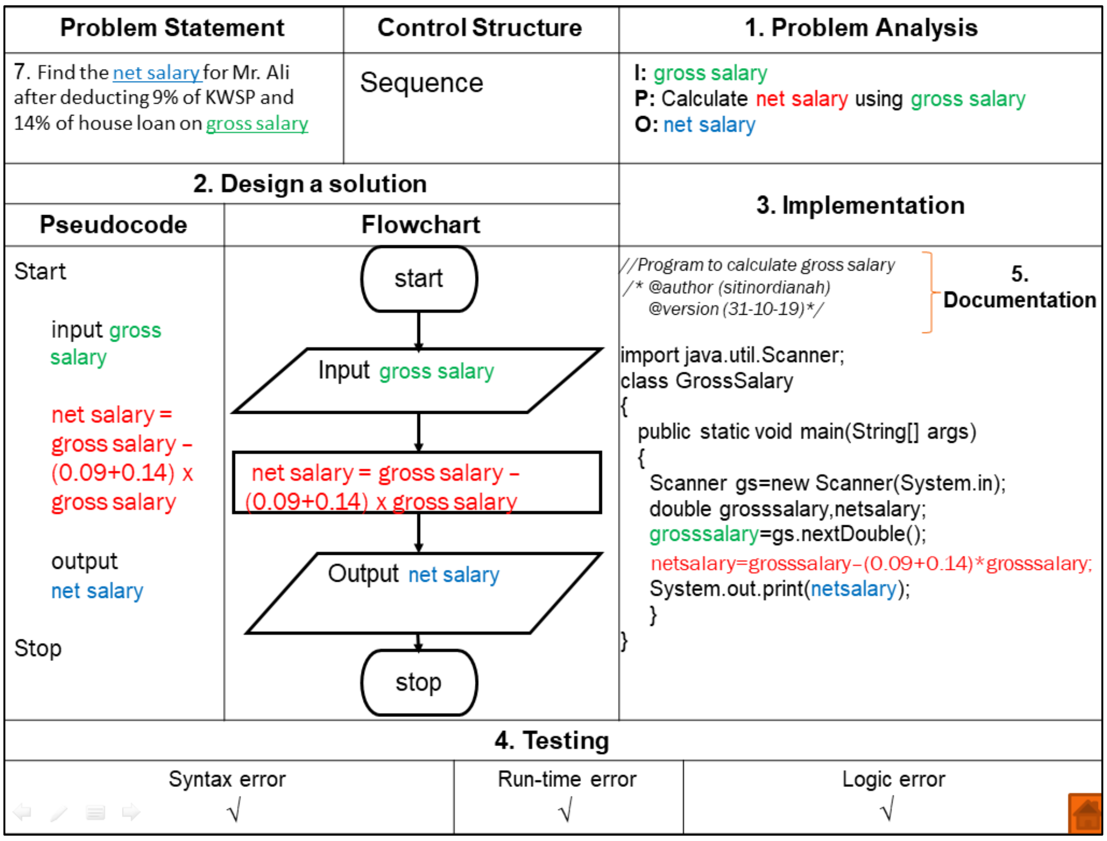
3. Implementation — **5. Documentation**

```
import java.util.Scanner;
class GrossSalary
{
   public static void main(String[] args)
   {
    Scanner gs=new Scanner(System.in);
    double grosssalary,netsalary;
    grosssalary=gs.nextDouble();
    netsalary=grosssalary-(0.09+0.14)*grosssalary;
    System.out.print(netsalary);
    }
}
```

4. Testing

Syntax error	Run-time error	Logic error
√	√	√

Figure 7: Step 4 Testing



Problem Statement	Control Structure	1. Problem Analysis
7. Find the net salary for Mr. Ali after deducting 9% of KWSP and 14% of house loan on gross salary	Sequence	**I:** gross salary **P:** Calculate net salary using gross salary **O:** net salary

2. Design a solution

Pseudocode

```
Start
    input gross salary
    net salary = gross salary – (0.09+0.14) x gross salary
    output net salary
Stop
```

Flowchart

3. Implementation

5. Documentation

```
//Program to calculate gross salary
/* @author (sitinordianah)
   @version (31-10-19)*/

import java.util.Scanner;
class GrossSalary
{
  public static void main(String[] args)
  {
    Scanner gs=new Scanner(System.in);
    double grosssalary,netsalary;
    grosssalary=gs.nextDouble();
    netsalary=grosssalary-(0.09+0.14)*grosssalary;
    System.out.print(netsalary);
    }
}
```

4. Testing

Syntax error	Run-time error	Logic error
√	√	√

Figure 8: Step 5 Documentation

Using this *EZ-Prog* technique, students know the relation between each step so that during the test, if the question ask the student to write either problem statement or pseudocode or flow chart or programming code, the student must identify the input, process and output from the problem statement and after that write the correct answer. The students can answer all the questions in the time given with minimal mistakes.

3.0 RESULTS AND DISCUSSION

Data analysis was performed by the number of correct answers and the percentage of grades in the pretest and posttest for 91 students. According to Embong (2009), the achievement of pupils can be measured by results in tests, in this case, the pretest and posttest. Therefore, a comparison of pretest and posttest was done to measure the effectiveness of the *EZ-Prog* technique.

Based on the pretest and posttest results, the findings of this study showed significant differences in achievement and interest in solving programming problems when compared to prior to using the *EZ-Prog* technique. In the posttest, the students showed interest in answering the questions related to solve programming problems. Students felt solve the programming problems was easy and the *EZ-Prog* technique was useful. After the *EZ-Prog* technique was introduced, students showed interest and were enthusiastic when the lecturer gave them a programming problem to solve. Students got excited to do an algorithm and program code using the *EZ-Prog* technique. During the study, all students' performance increased, inclusive of the slowest learner student.

Table 1 shows the difference in standard deviation and mean values between the pretest and posttest. There was a significant difference between using the *EZ-Prog* in problem-solving learning.

Table 1: Descriptive statistics between pretest and posttest

	n	Mean	Standard Deviation
Pretest	91	6.57	2.028
Posttest	91	7.86	1.488

Table 2 shows the results of paired t-test analysis. Paired t-test results revealed a significant mean difference t (12.226) = 90, p = 0.00. p = 0.00 (p <.01) for the difference in pretest and posttest scores.

Table 2: The result of paired t-test analysis

	T	df	Sig.
Achievements	12.226	90	0.00

This shows that the use of *EZ-Prog* adapted from Polya model as suggested by Malik and Jo-Coldwell (2017) and Wang and Hwang (2017) as one of the learning methods is an appropriate method of learning programming problem-solving. Besides, the results of the student response research also show that students can solve problem-solving and problem solving when both of these elements are important in the programming problem-solving process. Students' answers in each phase also indicate that they can understand the concept of programming before the encoding phase. In other words, the use of *EZ-Prog* can expose students to the concept of computing at an early stage. As reported by Stefania, Nikolas, & Vassilis, (2018), students face problems with mastery of programming languages because they are not exposed to computer features.

Finally, the positive findings from the use of *EZ-Prog* indicate that this technique can be used as a tool in teaching programming as it enhances problem solving skills among students. In fact, this method is not limited to matriculation students, but also all students that learn programming language and not limited to one programming language.

4.0 CONCLUSION

The use of *EZ-Prog* in the PDP helps to enhance students' problem-solving skills in programming learning. This *EZ-Prog* is an ideal approach for learning programming problem solving and can be used as one of the most advanced techniques by the lecturers.

ACKNOWLEDGEMENTS

Researchers would like to thank the Johor Matriculation College for their support throughout the research process. In addition, we would like to thank UiTM Shah Alam and UTM for their cooperation in sharing knowledge and encouragement in the success of this research.

REFERENCES

Agus, M., & Nanci, R., (2017). Pengaruh model Polya terhadap kemampuan pemecahan masalah matematika siswa kelas V SD. *International Journal of community service learning*, 1 (1), 31-38.

Embong M. N., (2009). *Keberkesanan Kaedah Konstruktivisme Terhadap Pencapaian Pelajar Tahun 5 Di Kuala Terengganu*, Terengganu.

Gagné, R. M. (1977). *Defining objectives for six varieties of learning*: American Educational Research Association.

Malik, S., I., & Jo-Coldwell, N. (2017). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22(3), 1089-1120. doi: 10.1007/S10639-016-9474-0.

Mayer, R. E., & Wittrock, M. C. (1994). Problem-solving. *Handbook of educational psychology*, 2, 287-303.

Olaniyan, A. O., Omosewo, E. O., & Levi, I. (2015). Effect of Polya problem-solving model on senior secondary school students' performance in current electricity, 3(1), 97–104.

Polya, G. (1945). *How to solve it: A new aspect of mathematical model*. USA: Princeton University Press.

Prahani, B. K., Limatahu, I., Yuanita, L., & Nur, M. (2016). Efectiveness of Physics learning material through guided inquiry model to improve student’s problem solving, 4(12), 231–242.

Stefania, S., Nikolas, T., & Vassilis, K. (2018). Effect of algorithms’ multiple representations in the context of programming education. *Interactive Technology and Smart Education*. 5(4), 230-243.

Wang, X. M., & Hwang, G. J. (2017). A problem posing-based practicing strategy for facilitating students’ computer programming skills in the team-based learning mode. *Educational Technology Research and Development*, 65(6), 1–17. doi: 10.1007/s11423-017-9551-0.

Widiana, I. W., Japa, I. G. N., Suarjana, I. M., & Diputra, K. (2018). The students’ ability to solve realistic Mathematical problems through Polya type problem solving learning model. *Journal of Education and Learning*, 12(3), 399–405. doi:10.11591/edulearn.v12i3.4526