SIT102 – Introduction to Programming

Answers for 8.MP: P Milestone 3 Checkpoint with Self-reflection

Milestone 3 – Weeks 6-8 More on Program and Data Structures

Part A – A quick self-check on your Pass tasks completeness

Have you addressed the given feedback from your OnTrack tutor to your on-time submissions of your weeks 6 to 8 **Pass** tasks?

Pass 6.1 6.2 waiting for 7.1 8.1 marks

Part B – Self-reflective Assessment

It is about time for your learning reflection. Based on your experience in SIT102 Milestone 3, please reflect on your learning using the following self-assessment table. For each row, place ✔ in the column which is most relevant to you.

Topics/Concepts	Have Some Doubts	Can implement using given guidelines	Confident to implement, apply, and justify
Static Arrays		✔	
Dynamic Arrays		✔	
Arrays with Custom Data Types		✔	
Loops and Indexing		✔	
UML Diagrams – Data Entity		✔	
UML Diagrams – Program Functionality		✔	
Other Languages – Data and Program structures		✔	
Others, please specify		✔	

Part C – Reflection Overview

Provide your reflection overview in 150 – 180 words for the above self-reflective assessment in regard to **various programming concepts and topics** in your **learning journey**. Try to reflect the following but not limited to:

- What was your understanding on the way(s) storing lots of data in a program? How different is it now? What programming artefact(s) could be applied?
- Recall what you have implemented so far, how do loops in a program help in managing lots of data?
- Imagine what would happen if we were not allowed to build a program with decomposition techniques? What would be the pros and cons? How could UML diagrams help in program design and implementation?
- After you have explored various programming languages, what are your comments on data and program structures across different languages?
- In general, what were the concepts/topics you were most confident in?
- What were your most interesting discoveries?
- Elaborate any significant programming milestones or hurdles you have overcome, and how could it be better.

Note: Your reflection in each milestone task are the building blocks for your learning summary report for your portfolio submission.

1 Most programs need data in order to work. Sometimes this data is provided to the program when it runs, and sometimes the data is built into the program. Variables are tools that help the programmer temporarily store data for a finite amount of time. Constants are tools that help the programmer define artifacts that are not allowed to change or make changes.
2 Loops are used when we want a particular piece of code to run multiple times. We use loops to execute the statement of codes repeatedly until a specific condition is satisfied. It eases the work of the programmer and also shortens the code length.
3 If we don't use decomposition techniques to build a program, the disadvantages are that the code becomes complex and it takes a lot of time to fix bugs if they occur. The only advantage may be to avoid using a bad design to implement the program, where problems can lead to incorrect analysis of the system if it is not designed correctly. **In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.**
4 the concepts for data structures and algorithms are same for all languages. The only thing that changes is the syntax **5 Loops and Indexing， Because I use them most often and in many programs。** **6 While Python is strongly influenced by C++, they are two completely different languages. Python is easier to learn but sluggish to run. C++ has faster, more efficient execution, but it can take years to truly master**
7 In loops and indexing, I didn't understand them at the beginning, including size i++, etc. After the teacher's explanation and self-learning, I gradually mastered them and expect to be able to use them in other programs

End of document (3 parts)