COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members	Sheikh Zaina	Youssef Bakr
Team Members Evaluated	ahmedt52	akrama7

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions. Completing the peer code evaluation on time will earn your team a total of **25 marks**. Do not exceed 2 paragraphs per question.

Peer Code Review: OOD Quality

1. **[3 marks] Examine the main logic in the main program loop. Can you easily interpret how the objects interact with each other in the program logic through the code? Comment on what you have observed, both positive and negative features.**

- The main logical loop is generally done fine and can be interpreted well enough. The interaction between the objects is well defined and clear to understand.
- Negative feature: There is one instance where cout is used instead of `MacUILib_printf`.
- Negative feature: Not enough comments
- Positive feature: User Experience
- Positive feature: Effective memory management

The negative features are inconsistency in the printing function and not sufficient comments. There is one instance where cout is used instead of MacUILib_print. The positive features include good user experience and good memory management. There is 0 memory leak.

2. **[3 marks] Quickly summarize in point form the pros and cons of the C++ OOD approach in the project versus the C procedural design approach in PPA3.**

Pros:

- The approach of using classes and objects makes it easier for the methods to be used in the project.
- It is easier to add new features.
- Improves readability

Cons:

- It is harder to debug in case of any logical errors.
- Memory management may be complicated.

Peer Code Review: Code Quality

1. **[3 marks] Does the code offer sufficient comments, or deploys sufficient self-documenting coding style, to help you understand the code functionality more efficiently? If any shortcoming is observed, discuss how you would improve it.**

The code seems to be well written with sufficient comments in certain files to understand what is going on. The comments play an effective role in explaining the functionality which makes it easier to follow along.
However, there is an inconsistency in the commenting style. Some parts lack comments entirely. We would overcome this by ensuring that the necessary comments are provided where deemed appropriate.

2. **[3 marks] Does the code follow good indentation, add sensible white spaces, and deploys newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.**

   Yes, the code follows good indentation, adds sensible white spaces, and deploys newline formatting for better readability. Each line is well-written and adheres to industry standards, making the code easy to read. However, I would suggest consistency in terms of the bracket placement.

Peer Code Review: Quick Functional Evaluation

1. **[3 marks] Does the Snake Game offer smooth, bug-free playing experience? Document any buggy features and use your COMPENG 2SH4 programming knowledge to propose the possible root cause and the potential debugging approaches you'd recommend the other team to deploy. (NOT a debugging report, just technical user feedback)**

   A bug free playing experience is offered to the users. However, there is a 0 being printed out above the game console that has no apparent use. The possible cause could be an unnecessary print statement or a logical error. Reading the draw screen function and/or using json would be a good approach to find the bug.

2. **[3 marks] Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause(s) of the memory leakage.**

   There is 0 memory leak.

Project Reflection

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:
[3 marks] Do you think the compound object design of objPos class is sensible? Why or why not?

- The compound object design of objPos class is sensible (to some extent) but it has pros and cons.

- Pros:
  - Allows for storing of x and y into 'one item'
- Cons:
  - Some methods are redundant and can be avoided by deploying a different coding approach.
  - If memory is not handled correctly, it can have catastrophic results.
  - Does not utilize encapsulation (all data members are public

[4 marks] If yes, discuss about an alternative objPos class design that you believe is relatively counterintuitive than the one in this project. If not, explain how you'd improve the object design. <u>You are expected to facilitate the discussion with UML diagram(s).</u>

Instead of creating an 'extra data type' through a struct, we can instead simply replace it with two integers (which as previously mentioned is generally less tedious as well as not having to deal with DMA in a similar way as the position pointer).

Another point to note is altering the symbol from being in the **public** area off the class to the **private** area. This is due to the concept of encapsulation in OOP (topic 12), which is not being used in objPos

Below lies the altered UML diagram for the objPos class:

objPos
- symbol: char - x_coord: int - y_coord: int
+ objPos() + objPos(x_coord:int, y_coord:int, symbol:char) + setobjPos(position:objPos) : void + setobjPos(x_coord:int, y_coord:int) : void + getobjPos():objPos const + getSymbol(): char const + isPosEqual(refPos:const objPos*) : bool const + getSymbolIfPosEqual(refPos:const objPos*) : char const