COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members	Haixiongtao Nie	Zihan Lin
Team Members Evaluated	Hermin3	Kapuri2

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 **30 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.
   - Comment is very descriptive, and easy to understand.
   - The draw screen logic can be placed inside the game mechanism class so that the main file can be cleaner and easier to understand.
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.
   - C++ OOD approach support parallel development which is more timesaving compared to procedural design approach.
   - However, OOD makes development process more complicated, it is very easy to make mistake and spend tons of time debugging for a new OOD programmer.

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 is easy to understand with enough descriptive comment.
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.
   The code is well formatted and easy to understand.

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)
   No bug or any problems during play time.
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.
   No memory leaks.

Project Reflection

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?
2. **[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.</u>

I do not think it is sensible. For a large-scale project, compound object design may optimise memory. However, in the small-scale project like the snake game, it makes the memory management more complex. In this project, we do not have lots of data, so frequently creating and destroying object makes this compound object design less efficiency.

To improve this, it is better to just define pos as a Pos Object. This reduces the frequently use of dynamic memory use and simplifies the memory management.

ObjPos
+pos:Pos +symbol:char
+objPos() + objPos(xPos: int, yPos: int, sym: char) + setObjPos(o: objPos): void + setObjPos(xPos: int, yPos: int, sym: char): void + getObjPos(): objPos const + getSymbol(): char const + isPosEqual(refPos: const objPos*): bool const + getSymbolIfPosEqual(refPos: const objPos*): char const