COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members	Nerar Hermiz Ishan Kapur. Group Hermin3 – Kapuri2
Team Members Evaluated	Jobs-and-Musk. MacID: Nieh5 1-member team

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.

   Ans- The main program loop is very clean and is extremely easy to read. In functions like drawscreen and runlogic the programmer makes it very evident that other objects are interacting with each other as there is almost no code in either of the mentioned functions other than calling other member functions from other classes. I believe that almost all of the member functions that were used in the main program loop were named in a way where their functions are intuitive, but the code could use comments to briefly explain what each function does just to make the code more understandable.

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.

   Ans- An advantage of using the C++ OOD approach over the C procedural design approach is that the code is way neater with class and object definitions in other files, making the code much more readable and easier to implement. It makes the code much more manageable and implementable. The Code can be reused through inheritance and polymorphism, reducing redundancy. A con of using this approach is that it makes the code more complex and leads to an increase of overhead runtime due to the creation of objects and methods. In PPA3, C procedural design approach was more favorable as it was a smaller project and was easier to implement.

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.

   Ans- There were a few shortcomings observed in respect to the commenting and self-documenting aspect of the code. There were no comments made in the actual code and hence it was a bit tedious to go through the code and analyze and understand it. Additionally, there were no comments on how to

exit the code mid game. However, there were certain instructions being printed out on the screen relating to the points of each character eaten which made it easier to understand and play the game.

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.

   Ans- The code was very well indented and the group added sensible whitespaces and the code was formatted quite well. The code was easy to read despite there being no comments and was neat. The code's readability could have improved by adding comments as they would have made understanding the code much easier.

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)

   Ans- The code runs flawlessly. They did the bonus parts too where each type of food counts as a different point as well as having multiple foods to choose from. Both of these extra functions work fine. Once a player runs into themselves, it prints out a losing message. However, when we quit the game there is no message that pops up other than to press enter to shut down. Additionally, it would have been nice to have the exit key printed on the terminal.

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.

   Ans- The code does not have any 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 your own, reflect on the following questions:

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

We believe that the objPos class design with the additional struct is useless because making the Pos struct outside of the objPos class is unnecessary as it makes the code repetitive. The objPos class already represents the x and y which makes the Pos struct not required. Additionally, we initiate symbol in in objPos but x and y in the Pos struct which doesn't make sense as those three components should be together. It makes it messy when they are scattered that way

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(s).</u>

Ans-We think that it would be easier if we simply incorporate the struct in the objPos itself and get rid of the Pos struct which is shown in the UML below. We deleted the struct Pos and we took its values (x and y) and initiated them in objPos instead of having it use it. We then would just need to change some minor things in the source code.