**COMPENG 2SH4 Project – Peer Evaluation [25 Marks]**

Your Team Members esmailm guagla3

Team Members Evaluated tsang8 girkik1

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.
   1. While observing the code, it was easy to follow the logic, as well as the integration of the various classes. The naming conventions for the varying parameters were intuitive. The order in which the methods are done also are intuitive, in addition to the interaction between the varying classes. For example, by initializing the board first and then adding the various food and player elements made this code quite efficient. Furthermore, by passing an object of the GameMechs class and the Food class allows for interaction between them.
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.
   1. Using the C++ OOD approach allows for more organized code, especially for bigger projects.
      1. For example, the run logic in PPA3 had an extensive amount of code. However, with an OOD approach, we can simply call functions of the Player class to do the procedures instead, providing a cleaner image.
   2. Using the C++ OOD approach means that we can dictate what classes are available to other ones, as well as what members in these classes are private or public.
      1. For example, an object of the Player class has access to an object of both Food and GameMechs, but food only has access to gameMechs.
   3. Using the C++ OOD approach means having multiple files for each class which requires a high level of awareness and discipline of all these files.
      1. For example, when changing the player from type ObjPos to type ObjPosArrayList, we had to be aware of how this would affect all the functions and mechanisms in multiple different folders.

**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.
   1. This groups code does not provide enough comments that would aid an observer in understanding the code. If someone who was unfamiliar with this project were to view it, it would probably not read as clearly. Additionally, various portions of the code were left commented out, including a previous attempt of the drawscreen logic. There was also a for loop that does not add the the functionality of the code. To improve on this, this group should focus on adding more comments to explain what is happening in the code, specifically in the main program.
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.
   1. This group utilized good indentation, as all loops where easily read and understood. Additionally, the group utilized white spaces well, adding them when starting a new block of code. Finally, in their drawscreen logic, newline formatting was implemented to improve the visual appearance of their game.

**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)
   1. This groups implementation of the Snake Game offers a smooth, bug-free playing experience. When using the WASD keys, the snake moved in its correct direction smoothly, as well as the rest of the snake body followed as expected. Additionally, when colliding with food, the snake length and score grew accordingly. Furthermore, the group incorporated clear instructions as to how to play the game. Finally, the game never once froze no matter how long the snake length got, showing that there is no segmentation fault.
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.
   1. When running Dr. Memory, this groups code showed 2 bytes of memory leak. However, this can be attributed to a compiler error. This means that all items stored on the heap were properly deleted after the end of the main loop. However, it was also found that there were 14 unique instances of 2408 invalid heap arguments. Though the cause of this is unknown, it could be due to several issues including deleting an instance that was never allocated on the heap, as well as double deletion of the same instance.

**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?
   1. After revising both ours and the other team’s implementation, we have concluded that utilizing the Pos struct was not a sensible design decision. As struct is not something often used in OOD, it would make more sense to make x and y public members of the objPos class. We believe this because symbol is already defined as a public member of the objPos class and is utilized in an almost identical way as x and y. It is also less convoluted to access these members, as when we need to call x and y for a specific instance of an object, we had to fist access it through the Pos struct instead of directly like symbol. For these reasons, we believe this design decision is not sensible.
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. You are expected to facilitate the discussion with UML diagram(s).
   1. As mentioned above, instead of utilizing a stuct, we can directly add the x and y positions of the object as a public member of the objPos class. As shown in the UML diagram, the position would now be defined by an int x and int y, alongside char symbol. Additional to this change, we would implement getX() and getY() to fetch the values of x and y, respectively, similar to that done for the symbol. Through this, we are able to make the implementation of the objPos class more intuitive. Now, when accessing the x and y values, we must call an object of the objPos class and then call the getter for either x or y, opposed to having to access the Pos struct initially.



**Figure 1.** UML diagram of the objPos class, featuring new public members x and y, as well as new public methods setX( ) and setY( ).