**COMPENG 2SH4 Project – Peer Evaluation [30 Marks]**

Your Team Members Justin Wei Sotheara Eng

Team Members Evaluated oreild6 woldeyed

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.

**Within the main program loop, it is very easy to interpret how objects interact with each other. The code is well commented, and the logic is easy to understand. Some positive features are the readability and simplicity of the code. However, there are some redundant codes throughout the project. One example of this was from line 120 to 130 where the code checks the food position only if the player isn’t printed there. This was done to prevent player and food overlap. This is redundant because their food generation function already guarantees the food doesn’t spawn on top of the snake. They also have a global variable that could have been implemented within the class (global variable collision could have been calculated within the movePlayer() function). Some minor negative implementations would be checking the lose flag within the clean up function where it should’ve been in the draw screen.**

1. **[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.

**As stated above, the OOD approach is better for large programs that deal with many complicated objects interacting with each other. It allows for the main program to be more like pseudo-code, which improves readability, and all the implementations can be dealt with separately within the class definitions. This isolation of functions also allow use to test them out separately without worrying about other parts of the code.**

**On the other hand, the procedural design approach in C used in PPA3 is a little easier to understand and wrap your head around. It’s better suited for smaller programs. Another downside is when organizing the roles and functionality of each class is a little challenging and can get overwhelming if one is not careful. This is especially true if some classes are like each other. Although in the pros, it was said that readability is improved within the main function, this isn’t true for the implementation of the functions where class member function reference and function references can get messy. An example would be within our draw screen function within the project.cpp draw screen function where a lot of arrow dereferences were called.**

**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 very well commented, and it was easy to understand. The comments are short, concise and do a very good job in breaking down the logic.**

1. **[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 presentation of the code of is very well done. The group does a good job using indentation, white spaces and new lines to make the code very readable.**

**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)

**The overall gameplay is smooth. However, there are a few bugs:**

* **Wraparound feature: When going down and wrapping back to the top wraps around one row too low.**
* **Wraparound feature: When the player is going down and reaches the bottom border. Pressing either left or right arrow keys will cause the player to disappear from the screen.**

This is most likely within the wrap around in the moveplayer() function in the player class.

1. **[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.

**The snake game has 208 bytes of memory leak. The memory leakage is likely due to incorrect implementation of copy assignment operator in objPos class.**

**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?

**I don’t think there are any problems with the design of the objPos class.**

* **The pointer to the struct pos makes the code more modular.**
* **The struct can easily be expanded to include more dimensions or more symbols.**
* **The pointers enable dynamic memory allocation.**

1. **[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.

**An alternative to the objPos design would just be instead of a struct you have x, y and symbol as private data members.**

**ObjPos**

**Private:**

- int x;

- int y;

- char symbol;

// The rest of the implementation remains the same