COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members: Veronica and Hafsa; najamh-ece, botrov1

Team Members Evaluated: guao31-lin422; lin422-compeng, Guoa31

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 interactions between objects in the program logic are straightforward. The game GameMechs, Player and Food objects communicate effectively making it easier to know what role is interpreted. The objects are interacting with each other through their methods rather than directly accessing data, for instance, Player relies on GameMechs to get user input and board size. Additionally, the use of debug keys (p, l and o) is a helpful feature for testing and troubleshooting. However, these keys can cause problems while playing the game because they are hardcoded directly into the players logic. For example, if a player accidentally presses one of the keys, it might increate the score or generate new food which could disrupt the game.

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.**
   The C++ OOD approach in this project has several pros and cons. It uses objects like Player, Food and GameMechs to manage their owns tasks which makes the code more organized. This also makes it simpler to update or add new features without breaking other parts of the code. Its great for bigger projects as each part of the program focuses on specific things which makes it easier. However, this approach is a lot more complex to understand and can use a lot more memory. Also, since objects are relying on each other it could limit the flexibility. In contrast, using C in PPA3 is simpler and focuses on functions and data directly which is a lot better for smaller projects. However, it doesn't really organize the code as well which an make it harder to code. It also relies on global variables which can cause unexpected issue when different parts of the program are affecting the same data.

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 provides a reasonable comment and she meaningful function and variable names which can help understand the functionality of the code. However, there is a lack of comments in more complex sections of the code which can be difficult to follow the logic or understand the purpose of those certain parts of the code. Overall, improving the commenting style would make the code easier to understand and maintain the code.

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 demonstrates a good indentation, ensuring a clear structure that helps read the code. It uses sensible white spaces and newline formatting makes the code more organized and visually appealing. To improve readability, for instance, some parts might have inconsistent spacing, or more blank lines could be inserted in between larger logical blocks, like separating conditionals, loops, or declarations. Logical sections should be visually divided with blank lines to improve the overall structure.

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)

   Yes, the game does offer a bug-free playing experience, with smooth snake movement and wraparound mechanism. The snake death and forced exit implementation is correct. Overall, there are no buggy features in the program.

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.
   The snake game does not contain any memory leakage. The game successfully handled memory deallocation resulting in 0 bytes of leakage.

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?**
   The dynamically allocated Pos struct is part of the objPos class's compound object design, which is functional but not sensible for the circumstances at hand. The design adds needless complexity even if it adheres to Object-Oriented Design principles by combining position (x, y) and symbol into a single, coherent unit and guaranteeing appropriate memory management with deep copying. Since x and y could just be member variables of the objPos class, there is no need for the dynamic allocation

of the Pos struct. This would make the code simpler and less prone to errors by doing away with the requirement for explicit memory management, as seen in the destructor, copy constructor, and assignment operator.

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).