COMPENG 2SH4 Project – Peer Evaluation

Your Team Members __________Yasandu Gunawardana__________
______Abdul Moeez Bhatti______________

Team Members Evaluated _________Abdul Wahab___________
_______Amadou Wane_____________

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

1. **[6 marks]** OOD is about sensible code modularization. Looking at the header files of each object, can you easily interpret the possible behaviours of the objects involved in the program, and how they would interact with each other in the program? Comment on what you have observed, both positive and negative features.

- **Positive Features:**
  - Header files provide a clear separation of interface and implementation for each object (e.g., GameMechs, Player, objPos, objPosArrayList).
  - Object behaviours can be inferred from the class definitions and member function declarations.
  - The use of class constructors and member functions promotes encapsulation and abstraction.

- **Negative Features:**
- There are minimal issues with the code in terms of code modularization. However, providing additional comments or documentation in the header files would enhance the clarity and ease of understanding for other developers working with these classes.

2. **[6 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.

Positive Features:

- The main program logic is well-structured and easy to understand.
- Objects (GameMechs and Player) interact seamlessly to control the game flow.
- The separation of concerns between game mechanics (GameMechs) and player behaviour (Player) is clearly defined, promoting modularity.
- Input handling, game logic, and rendering are distinct and logically organised within the loop.

Negative Feature:

- There are no significant negative features observed in the main program logic. It effectively demonstrates the benefits of object-oriented design, making it clear how objects interact with each other to implement the game's functionality.

3. **[5 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.

Pros:

- Modularity: The C++ OOD approach encourages modular design, allowing for the separation of concerns and improving code organization.
- Readability: Object-oriented code tends to be more readable and understandable due to its focus on modeling real-world objects and their interactions.
- Encapsulation: Objects encapsulate data and behavior, enhancing data security and reducing the risk of unintended side effects.
- Reusability: Object-oriented code promotes code reuse through inheritance and polymorphism, reducing redundancy and maintenance efforts.
- Abstraction: Abstraction simplifies complex systems by providing a high-level view of objects and their interactions

Cons:

- Learning Curve: Object-oriented programming can have a steeper learning curve for those new to the paradigm compared to procedural programming.
- Overhead: In some cases, OOD can introduce overhead due to the abstraction and encapsulation layers.
- Complexity: Overuse of object-oriented concepts can lead to overly complex designs, making the code harder to understand and maintain.
- Performance: Object-oriented code may have slightly worse performance compared to well-optimized procedural code in certain scenarios.
- Design Overhead: Developing a well-structured object-oriented design can require additional planning and design work upfront.

Part II: Code Quality

1. **[5 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 demonstrates a good level of professionalism in its structure and organization. It is generally well-structured, making it easy to follow the flow of the program. Variable and function names are mostly descriptive, contributing to code readability. While comments are present and helpful, there's room for slight improvement in terms of providing high-level comments for classes and more detailed explanations for complex logic sections. Overall, the code maintains a professional standard, and with a few minor enhancements in documentation, it would be even more accessible to developers.

2. **[4 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 good indentation, sensible white spaces, and employs newline formatting effectively, which significantly enhances its readability. It follows a well-organized structure, with proper separation of functions and code blocks. This clear structure makes it easy to navigate and understand the code's logic. The consistent use of naming conventions and meaningful variable names also contributes to its clarity. Overall, the code is easy to read and comprehend, promoting better maintainability. However, some additional comments or documentation could further improve the code's understanding, especially for someone new to the project.

Part III: Quick Functional Evaluation

1. **[8 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.

2. (NOT a debugging report, just a technical user feedback)

   The Snake Game implementation runs smoothly without any noticeable bugs or issues. It offers a hassle-free and enjoyable gaming experience. The code appears to have been thoroughly tested and debugged during development, resulting in a well-executed project. It demonstrates a good understanding of programming concepts and practices. The code is well-structured and easy to follow, contributing to its overall quality. Overall, the Snake Game project appears to be free of significant bugs or issues, and it doesn't require any immediate code changes or debugging efforts from the other team. The game runs smoothly and offers a seamless playing experience, which reflects the high-quality implementation and attention to detail in the code. It's a commendable achievement, showcasing a well-crafted game that players can enjoy without encountering any disruptive problems.

3. **[6 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause of the memory leakage.

   The code ensures proper memory management through the cleanup function. In the CleanUp() function, the allocated memory for both the Player and GameMechs objects is explicitly deallocated using the delete operator. This approach ensures that all dynamically allocated memory is released, preventing memory leaks. Therefore, there are no memory leaks in the code because the developers have appropriately handled memory deallocation in the cleanup process

Part IV: Your Own Collaboration Experience (Ungraded)

1. Tell us about your experience in your first collaborated software development through this project – what was working and what wasn't. If you are a one-person team, tell us what you think may work better if you had a second collaborator working with you.

   Certainly, the collaborative software development project involved a partner and everything went smoothly, it's likely because my partner and I split the work evenly, following a 50/50 split. This balanced distribution of tasks and responsibilities can contribute to a harmonious collaboration. It ensures that each team member has a clear understanding of their role and can focus on their assigned areas without overburdening one another. Effective communication and synchronization between partners would also be key to achieving this successful outcome.