COMPENG 2SH4 Project – Peer Evaluation

Your Team Members __Juvani Karunabalan & Rebeca Gaston____

Team Members Evaluated __Audrey & Alex______

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

Part I: OOD Quality

1. **[5 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.

   Looking at the header files of each object, the code is easily interpretable in terms of object behaviour. The member functions are named in a self-explanatory manner. In addition, Player header file was commented thoroughly to ensure that the reader can understand the purpose of each member. However, line 9 in Player.h has an unnecessary declaration of the food class, which should have been omitted.

2. **[5 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.

   In terms of the main logic, there were mainly positive features of the code. Global pointers were clearly labelled at beginning of the code. The main program is easy to interpret, the logic is ordered in a proper manner. OOD was implemented correctly due to the easy readability of the global pointers accessing function members of distinct classes. Most of the logic was properly separated into their respective functions. However, in the main program, the void getInput(void) function was left empty. The Game Mechanics pointer to getInput() should be included in this function for readability, even if the program executes without it.

3. **[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.

   Pros :

   - C++ OOD approach is much more modular compared to the procedural design approach
   - More organized
   - As a result, it is easier to interpret

- Not overwhelmingly long (in terms of blocks of code)
- Less use of Global variables
- Declaration of variables are less scattered
- More improved Manageability and Reusability (ex. We are able to utilize inheritance to avoid copy and paste code)

Cons :

- Too many files to search through when debugging
- Takes longer to compile and debug compared to procedural programming in C

Part II: Code Quality

1. **[4 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 program logic is ordered in an organized and logical manner. Comments were balanced and straightforward, at least having one comment/explanation per function. However, if someone were to read this code without having prior background experience of the project, more line-by-line comments should be implemented to explain larger blocks/loops of 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.

   Most of the code/comments were indented and spaced out properly. Only one header file (Food.h) had faulty indentations that did not correlate with the rest of the formatting.

Part III: Quick Functional Evaluation

1. **[6 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 a technical user feedback)

The Snake game executed in a bug-free manner, and all contents are printed on the screen properly. Lose flag and exit key are differentiable and work as intended, both printing the right score. Wrap around and snake movement is correct. The increment body/score features are working as intended when a food object is consumed, and food is also randomly generated as expected.

2. **[4 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.

Snake game has no memory leak.

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.

Iteration 1 and 2 were separated adequately in terms of workload. However, it was not clear as to how we could've split up iteration 3 in an equal way. This iteration had three features where the first two were considerably harder than the third one. This resulted unequal workload. In addition, one partner would have struggled understanding how to continue off of their partner's work from the first 2 features of iteration 3.