COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Sanna Ghai and Abbiraa

Team Members Evaluated Maliha and Mariam

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.

The code was provided with adequate comments and simple logic to interpret the behaviours of the code and how they interact with other. It was easy to understand how the code was developed and why they used the logic they used for their functions. However, the exit flag was slightly unclear to interpret as it differed from the PPAs and was a challenge to interpret. [5.5/6 marks]

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.

The main loop had a clear flow of control as it enhanced the readability and making it easier to understand the sequence of the operations and functions. The main loop also effectively manages the dynamic state of the game by updating the player position, user input and generating the food position. [6/6 marks]

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.

In the C++ Object-Oriented Design (OOD) approach, using classes like GameMechs and Player helps keep things organized and easy to understand. The dynamic updates to the game state during the main loop make the game flexible and adaptable, and the focus on code reusability means you can easily add new things later. However, there are challenges, like using global stuff that might make the code too connected and mixing up drawing and game rules in one function, which can get a bit messy. On the flip side, the C procedural design in PPA3 is straightforward and easy to follow with a direct flow of control. But, it might get tricky to make changes, and you could end up writing the same code over and over, making it hard to fix and improve the program in the long run. [5/5 marks]

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 offers a sufficient number of comments for each function that either describe what the function is or what it does. Further comments are added in explaining the logic of a few lines, allowing for the user to easily understand the code. No shortcomings are observed in this section. [5/5 marks].

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 is very well written in terms of formatting. Each file follows the same formatting in terms of indentation, brackets are placed in a neat and consistent manner, and newlines are added between functions, allowing for better readability. No shortcomings are observed in this section. [4/4 marks].

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

This group has unauthorized modification in their MacUI library configuration leading to compilation errors on other platforms.

MacUILib.c

```c
#include "MacUILib.h"
#include <ncurses.h>

print_ptr MacUILib_printf;

#ifdef WINDOWS

    #include <conio.h>

```

As seen here, line 2 has caused this file to corrupt on users that are not Mac or Linux, causing code to not run or compile on Windows users. However, once the code has been changed to Window's users, the code runs smoothly and provides a bug-free experience, with key instructions to control the snake and end the game. [6/8 marks].

2. **[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.

```
~~Dr.M~~        0 unique,     0 total warning(s)
~~Dr.M~~        0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~        0 unique,     0 total,      0 byte(s) of possible leak(s)
```

No, this snake game does not cause a memory leak. As seen here, when drmemory executes there are zero bytes of leaks as well as zero bytes of possible leaks. [6/6 marks].

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.

   Overall, we had an enjoyable experience working together on our first collaborated software development. Working as team, allowed us to dedicate more time to this project based on our individual schedules, leading to a better overall result. Furthermore, it enabled us to expand our coding knowledge and skills and learn from each other. We were also able to split up the iterations, allowing us to individually focus on the functions at hand and achieve improved results.

   One obstacle we did face was difficulties in sending the project files back and forth. When attempting to send these files we experienced many issues as various ways to send the files were blocked due to security reasons. We found that sending the folder on teams or on discord were the only two methods that worked. Further issues arose when executing the project on a different laptop as that often created additional errors which needed to be fixed before trying to execute again. Overall, we found the best way to collaborate was to work in person, on one laptop. This method not only eliminated the need for files to be transferred, but further allowed for increased communication in our team.