COMPENG 2SH4 Project

Our Team Members:	laeeqm-400445935	khana421-400399089
Other Team Members:	ahmem121-400512517	yangj351-400507616

OOD Quality

1. When examining the code in the main project file, it is noticeable that our peers initialized a few global variables such as bool exitFlag and char input, which should not have been the case for this project.
   However, the rest of the code is well written out and the objects can be easily interpreted in how they interact with each other in the code. There are very minor typos in the code such as a misspelled word, or lack of a comma, but nothing that causes confusion.

2. Pros:
   - Using classes like Food, GameMechs and Player makes it easier to incorporate and include functions and invoke them, making the code more modular, and can be used for other projects.
   - Using private and public members in classes allows for better control over the state of objects used in the game, making it easier to create changes without creating unwanted behaviours or modifications.
   - It allows for better use of adding more features to the game, without having to rewrite large portions of code to run.
   - Makes debugging and code understanding much easier, with the separation of objects. Only need to focus on certain parts of the code instead of looking at the entire code related to the project.
   - Usage of constructors and destructors, member functions, and operator overloading also makes the code more friendly and enhances the coder's ability to read through without difficulty.

   Cons:
   - Too complex when it comes to defining several classes, managing several files, and more.
   - Know how to properly allocate dynamic memory and then how to properly deallocate and delete the memory to prevent memory leaks.
   - Debugging may be harder to navigate since there are several classes and members to go through when debugging.

Code Quality

1. When reading through the code, we noticed that the code has sufficient comments which makes it very easy to follow what the authors achieved from a specific line. Additionally, the code itself is well written in the aspect of self-documenting coding style so that you can understand what is

being called, pointed to, or evaluated. These combined to make sure that the code was efficiently understood and made sense every step of the way.

One thing we would suggest though is that regarding their commented out some parts of their code, as they had updated certain functions or removed certain members that were not needed, in their final submission. We suggest that they should have just deleted it completely as opposed to commenting it out, so as to not cause any confusion or potential errors if they were to appear. This is not an indication that anything was bad on their part, solely a better coding practice if you ask me.

2. The code does indeed follow good indentation and very sensible white spaces. When running the code to display the game, new lines are made to write important information, and there are reasonable connections when there isn't a need for a new line such as incorporating all types of food on one line, regardless of type.

   Although this isn't necessary, and it can be deemed more of a personal preference type of scenario, we would have preferred that the opening brackets were consistent. For example, some if statements or for loops have the opening bracket in the expression line, while other instances have the opening bracket underneath the loops/statements. Again, this is simply a matter of readability, which allows the editor to follow a specific convention making it easier to differentiate between nested loops/statements.

Quick Functional Evaluation

1. The snake game developed by our peers is very smooth and offers no errors or bugs when running. There are no bugs present when the game is being played, and there are no issues when leaving the game. Thus, there is no reason to debug anything.

2. When checking for memory leakage, we noticed that the game does not have any leakage, with a total of 0 total leaked bytes. Thus, there is nothing to report or identify of any memory leaks.

Project Reflection

1. The compound objPos class is a good way to represent an object on the screen. The issue, or rather, the redundancy comes from the pos struct. The pos struct only contains x and y,whereas the objPos contains the pos struct and the symbol for the object. We could combine these two to make one class, with that new class being called objPos. The new objPos class would have three private data members, a pos_x, a pos_y, and a symbol. This would reduce the redundancy caused by the pos struct.