COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Anna Dienstmann & Remsha Akbar Hussain
Team Members Evaluated	Darren Temporale & Konstantin Delemen

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.**

   Upon inspecting the header files of each object, it is fairly easy to interpret the possible behaviours of the objects involved in the program. This group decided to implement a separate food class rather than incorporating it into the GameMechs class. This approach enhances code modularization since functions pertaining to similar aspects of the program are grouped together, making it easy to access and evaluate specific classes which may be causing an issue. For example, if there was an issue with food generation, that separate food class will allow for easier debugging and help to pin-point what exactly is behaving incorrectly as it is isolated. Implementing a separate class helps with organization and can be more intuitive to a viewer since it is strictly dealing with food-related functions and data.
   Another positive aspect of the header files was the clarity of the functions contained within them. Each function was defined with its type (ex. bool, void, etc.) and its parameters when applicable. This allows for any other programmer to easily be able to assess the behaviours of each object and how they are utilized.

   One feature which could be improved upon within the header files are some repeated names for functions. For example, as below, the GameMechs.h file contains two functions named "incrementScore", and although they are differentiable since one has a parameter, upon initial inspection of the code, it may be difficult for another editor to differentiate amongst them and create confusion as to how the score is behaving.

   ```
   void incrementScore();//increment the score by 1
   void incrementScore(int val);//increment the score by val
   ```

   Overall, this group's header files were well structured with private and public functions and made it easy to interpret how the objects will behave.

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.**

   As this program contains several objects, when examining the main program loop it can be noted that the main loop is very concise and easy to interpret. A positive aspect which ensured the main program loop was easy to interpret was the accurate usage of references when accessing functions from other classes. As the functions were named intuitively, it made it easy to understand how certain objects would be behaving in the main loop. Similarly, variables created in the main loop are also assigned names which are easy to interpret. Within the global scope, three pointers are declared with clear indication of the class they are contributing from. For example, (Food *food) is representative of the food object being utilized. When creating new player and gameMech within initialization, the objects utilized are passed as an argument. This is representative of which functions will be utilized for the newly created gameMech and player, indicated by the object they pertain to.

   A positive decision was to move the drawScreen logic into a function in the GameMechs class. This cleaned up the main program file significantly compared to ours. Its placement within GameMechs also allows for easy access to certain variables.

   A positive stylistic feature was to print a message explaining the controls of the game and the rules. Although adding or removing this does not change anything logically in the main loop, it acts as a method of enhancing the users' experience when playing the game.

   A feature which can be improved upon is the structure of the GetInput() function. In their main loop, the GetInput() function contains an if condition for retrieving keyboard input. In order to make the main program more concise, this if condition can be implemented in the GamesMech class since it is a preexisting function of that class. The approach they have taken is not incorrect and still produces the same result, however this suggestion increases code modularization.

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.**

   C++ – OOD Pros / C – Cons:

   - C++ Encapsulation allows data privacy and control through specifiers (private, public, protected)
   - C++ Inheritance- more convenient
   - C++ Polymorphism- more convenient
   - OOD compatible with many languages

   C++ – OOD Cons / C – Pros:

   - Takes longer to construct code for OOD
   - Navigation more complex with OOD
   - C procedural formed in a sequence- helps with readability and bugs may be easier to catch
   - Procedural allows global data- convenient

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 is well-commented and very organized, indicating a sufficient self-documenting coding style. At the start of every loop or decision there is a comment explaining the purpose and functionality, making it easy to interpret what is occurring. The variable and object names are also appropriate and easy to understand. One note to be made is that there are spelling errors in the comments, however none subtract significantly from the idea they communicate. The following is the strongest example of this shortcoming.

```
if (xtemp == listEle.x && ytemp == listEle.y) {
    //If the randomly generated x and y valuse match the previous foop positions, repete the do while loop
    match = 0;
```

   A recommendation would be to proofread the comments before greenlighting the code, so that potential readers have an easier time deciphering the functions of the program's components.

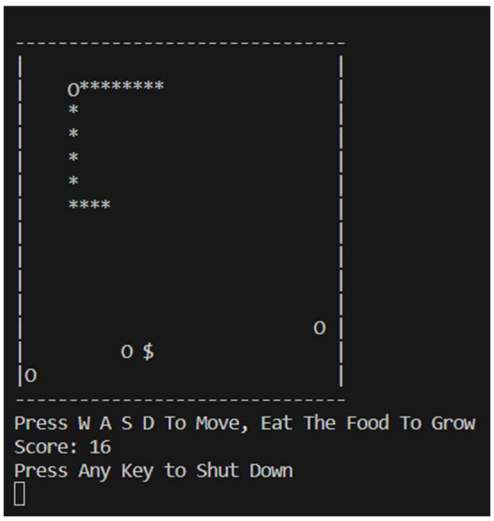
2. **[4 marks] Does the code follow good indentation, add sensible white spaces, and deploys newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.**
   One thing that could be observed is the placement of curly braces around loops and functions.

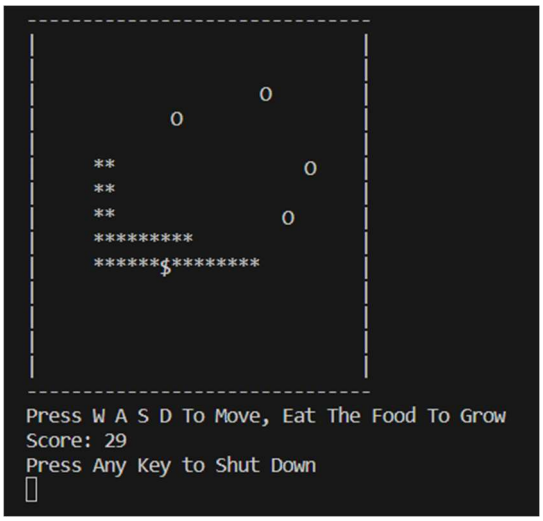
```
            objPrinted = 1;
        }
    }
    if (objPrinted == 0) { //If we havent printed a food object,
        for (int k = 0; k < (snakeLoc->getSize()); k++) {
            snakeLoc->getElement(tempSnakeEle, k);
            if (i == tempSnakeEle.y && j == tempSnakeEle.x) {
                objPrinted = 1;
                MacUILib_printf("%c", tempSnakeEle.symbol);
            }
        }
        if (objPrinted == 0) { //if we still havent printed anyth
            MacUILib_printf(" ");
```

   While the way they format it is more space-efficient, programmers in a professional setting will prefer curly braces to have their own line because it enhances readability for the next editor. Both styles appear in their code--it would be a good idea to at least remain consistent with the choice of that formatting.
   Otherwise, the program has consistently excellent indentation and spacing practices. We spotted only two offset lines in the whole program.

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)**





One bug we identified was the occasional overlaying of a food item onto a coordinate the snake's body should appear after an O is eaten and the foods respawn. This indicates a semantic bug either in the drawing function or the food generation function. Looking over their loops for food checking the snake body's coordinates, no errors are immediately visible. The most likely is a line for generation being placed before the calling of a checking function when the player updates, that only causes a problem in rare cases.

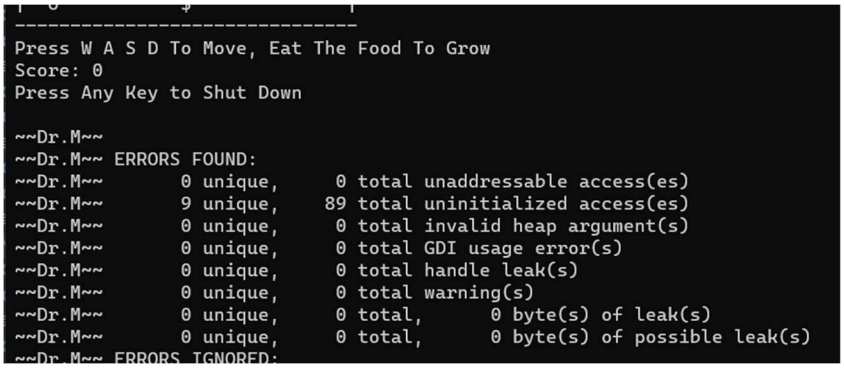
A potential debugging approach would be to set the gameboard to smaller dimensions like 16 x 8 and configure the food generation debugging key to generate all five pieces of food at will. This would increase the likelihood of the error appearing. Then, tweaking things such as the snake body list size or food bucket size to observe in what circumstances the bug appears. Using a debugger tool, they could also check the variables such as xtemp, ytemp, match, and the player body coordinates at every loop of the random generation function to see how they update.

```
51              if (xtemp == blockOff->x && ytemp == blockOff->y) { //If the randomly generated x and
52                  match = 0;
53              }
54              for (int j = 0; j < i; j++) {
55                  returnPos->getElement(listEle, j);
56                  if (xtemp == listEle.x && ytemp == listEle.y) {
57                      //If the randomly generated x and y valuse match the previous foop positions,
58                      match = 0;
```

Besides this, the program runs very smoothly and offers a pleasant player experience.

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.**

As shown below, this snake game had 0 bytes of leakage. They did an adequate job of deallocating the heap memory when appropriate—as class deconstructors and in the void CleanUp function in main.



This is good.

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.

   Anna:
   Having a team for this project was definitely beneficial for the pace at which it was completed. The recommended workflow outlined in the manual also helped split the tasks in an even way between groupmates until the last iteration. This was needed because VSCode doesn't allow for live collaboration between programmers. At least, not by a method we were informed of. I'm glad I was able to choose my teammate because if I had known no one in the course, or if my partner was randomly assigned, it would have been harder to trust and communicate with them, and the process would not have gone as smoothly.

   Remsha:
   I really enjoyed my first collaborated software development project, and the process went very well. Working with my partner was incredibly helpful in several different aspects. Designating different roles amongst us definitely made this project more manageable. This was also a great learning experience in terms of bettering my programming skills and I was able to learn a lot from my partner by gaining a fresh perspective. In order to ensure everything went smoothly, my partner and I made sure to consistently communicate with each other. One thing which was a little difficult was navigating a way for us to collaborate on the final iteration since I do not live in Hamilton, however we were able to find a method which worked best for us and efficiently completed the project. Overall, I gained a lot of new knowledge through this project and pair programming was very beneficial.