COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members Manav Patel + Dhruv Bagga

Team Members Evaluated iBioDreamTeam _____

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions. Completing the peer code evaluation on time will earn your team a total of **30 marks**. Do not exceed 2 paragraphs per question.

Peer Code Review: OOD Quality

1. **[3 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 logic in the main program loop function is well-organized and easy to comprehend. Each object, like Player, GameMechs, and Food, has a clear purpose. The way positions are checked in DrawScreen() could be more efficient, and switching to smart pointers for memory management would help avoid potential issues. Overall, it's a good start, but some adjustments could make it cleaner and more flexible.

2. **[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 of C++ OOD:

- Classes can be reused without changing existing code (eg. objPos was used for snake and food features)
- It is easier to locate and fix bugs or make changes to specific aspects of code
- Code is more organized and easier to read
- Able to scale up the project a lot easier

Cons of C++ OOD:

- Writing the code itself is more complex due to inter-object communication, data management and overall set up

Pros of C procedural design:

- Easy to write and understand for small programs (which the final project is not)
- Code is linear and works step by step down the file

Cons of C procedural design:

- Hard to scale because the code is not modular
- Lots of code duplication even with the use of function

Peer Code Review: Code Quality

1. **[3 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.

   Yes, the code offers more than enough comments, you can understand what is going on in the code. There was no shortcoming as your group checked all the boxes.

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.

   The code overall follows good indentation and formatting practices, for example, even in the drawScreen function where there are long lines of code and is a little busy, the program maintains its readability using indentations. However, to reduce clutter, there could have been more effective use of whitespace in the program as the code felt compact and a little difficult to read at points. Moreover, excessively commenting sometimes does negatively impact this, we do think they were a little overused which impeded readability, once again in the drawScreen function. However, this was a part of the project, so their overall impact is negated.

   There was also the method clearArray() in objPosArrayList that had commented out code that wasn't a part of the functionality of that method. This didn't effect the performance of the program, but added some unnecessary clutter.

Peer Code Review: Quick Functional Evaluation

1. **[3 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 technical user feedback)

   The snake game is smooth and bug free with funny exit messages, it is very well developed! All features worked as expected, and the bonus feature was very effective. No errors in wraparound, player growth, point system or movement.

2. **[3 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause(s) of the memory leakage.

   Testing through drmemory, we found no memory leakage in the team's code. Through further inspection, the cleanup function worked as intended, and every time "new" was called, there was a proceeding "delete".

```
SUPPRESSIONS USED:

ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
     10 unique,   544 total uninitialized access(es)
      0 unique,     0 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      0 unique,     0 total,      0 byte(s) of possible leak(s)
ERRORS IGNORED:
```

Project Reflection

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

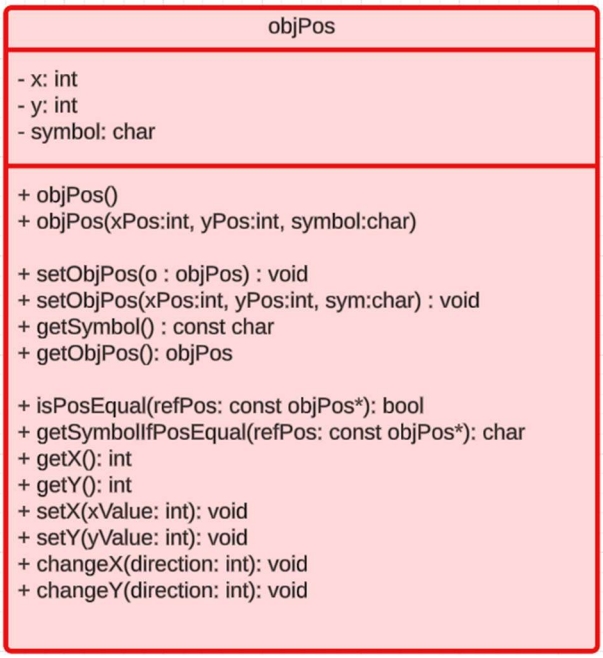
Although separating the Pos struct applies an additional modular concept to the code (and have the functionality of separating each individual coordinate on the game board into its own separate data), it is unnecessary to have used it in this assignment. It adds an additional layer of complexity when x and y could have just been put into the objPos class as private data members. For example, the Pos* pos pointer that calls on the struct can increase the potential of running into memory leak. Another example of this structure for the program causing unnecessary complexity is in the objPos methods (getObjPos() for example).

```
objPos objPos::getObjPos() const
{
    objPos returnPos;
    returnPos.pos->x = pos->x;
    returnPos.pos->y = pos->y;
    returnPos.symbol = symbol;

    return returnPos;
}
```

2. **[4 marks]** If yes, discuss about an alternative objPos class design that you believe is relatively counterintuitive than the one in this project. If not, explain how you'd improve the object design. You are expected to facilitate the discussion with UML diagram.

To improve the design of the objPos class, we would remove the Pos struct and store x and y as integer private data members in objPos. From here getter and setter methods can be used to access the x and y members just like how we have it set up for the symbol member. This negates the need for DMA as we are directly working with the private members instead of pointers to the struct. Here is a UML to demonstrate the design change.


objPos
- x: int
- y: int
- symbol: char
+ objPos()
+ objPos(xPos:int, yPos:int, symbol:char)
+ setObjPos(o : objPos) : void
+ setObjPos(xPos:int, yPos:int, sym:char) : void
+ getSymbol() : const char
+ getObjPos(): objPos
+ isPosEqual(refPos: const objPos*): bool
+ getSymbolIfPosEqual(refPos: const objPos*): char
+ getX(): int
+ getY(): int
+ setX(xValue: int): void
+ setY(yValue: int): void
+ changeX(direction: int): void
+ changeY(direction: int): void