COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members maklaa1@mcmaster.ca jensenjj@mcmaster.ca

Team Members Evaluated ghosha20@mcmaster.ca manojv3@mcmaster.ca

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 **25 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 code demonstrates a solid implementation of object-oriented design principles. The game's key components include player movements, food management, and game mechanics. The interactions between objects are straightforward. They explained how to escape the game and when you die, show a text explaining that you lost with your final score.

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 the C++ OOD Approach:
- Classes keep data together, improving modularity and readability.
- **P**romotes code reuse through classes and objects, simplifying future enhancements
- Clear separation of responsibilities among objects makes the system easier to understand

Cons of the C++ OOD Approach:
- Object-oriented code can introduce more complexity
- Increased risk of memory leaks if not handled carefully.

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.

Code was set up well and written in sections. There was a comment in each section explaining what it does so it was easy to understand. No problems found with code functionality.

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.

Code was easy to understand, and all follow good indentation making the program structure easy to follow.

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)

Upon review, the code does not display any bugs and runs very smoothly. The code is bug free and the Snake Game runs perfectly fine.

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.

No, according to the memory profiler, Dr. Memory, there are no memory leaks when running the Snake Game.

Project Reflection

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

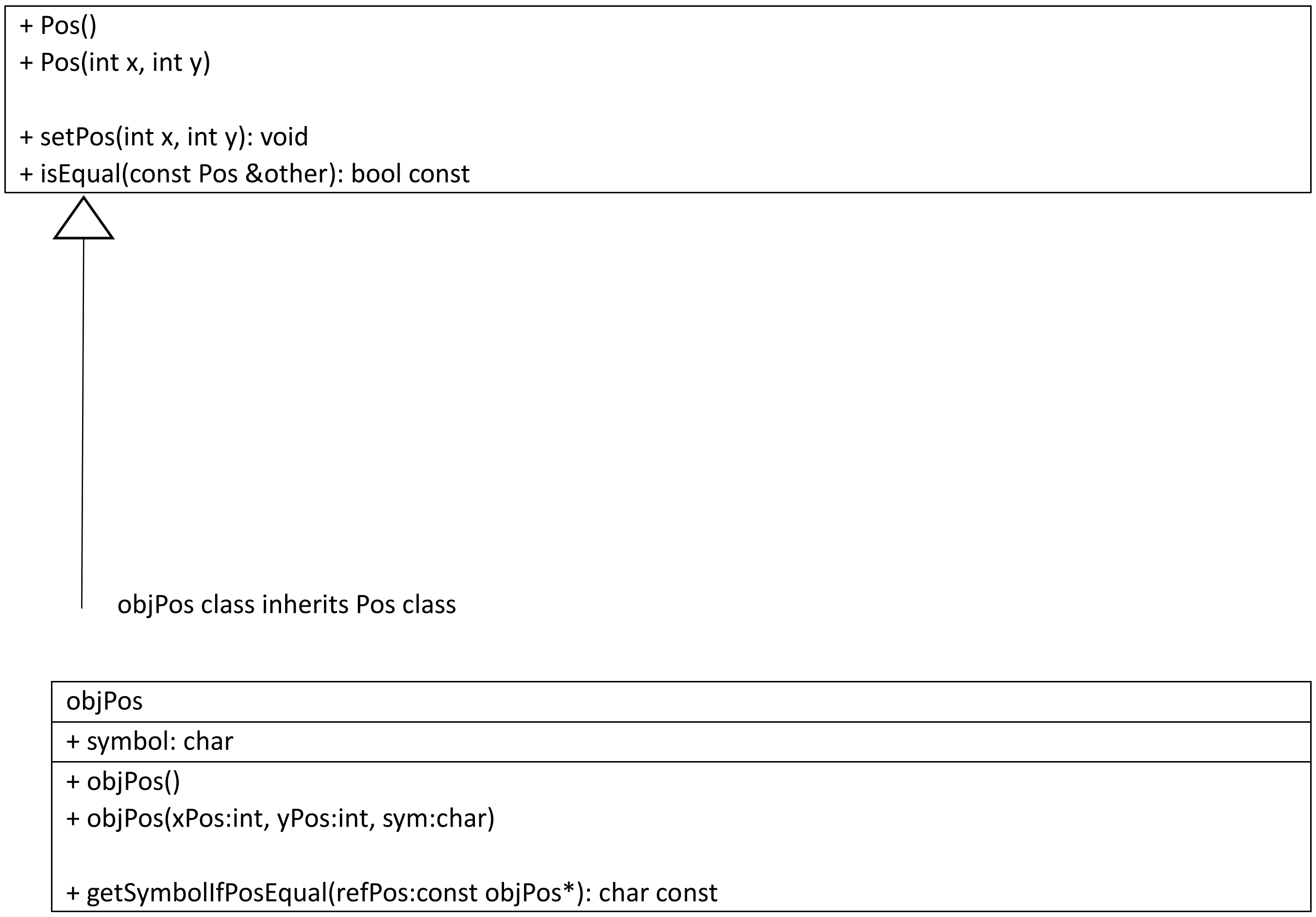
1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

I think this implementation of objPos class works just fine for this project, but it may not be conventional in an OOD environment. The reason is that Pos could have been defined using an object in objPos, and this would better blend into the code, which is object oriented as a whole. However, I do think that the implementation of objPos is sensible as it makes accessing accessing the position of the object more straightforward and easier to understand, especially for newcomers to C++ and OOD.

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. <u>You are expected to facilitate the discussion with UML diagram(s).</u>

I do think the current design is sensible. A counterintuitive implementation would be to implement Pos as a class of its own, complete with a .h and .cpp file. Then, the objPos class would inherit the Pos class. This would make accessing objPos and Pos much more difficult and tedious than it needs to be, thereby complicating the code and making it more error prone. Below is a UML diagram to illustrate this alternate implementation.

Pos
+ x: int + y: int



```
+ Pos()
+ Pos(int x, int y)

+ setPos(int x, int y): void
+ isEqual(const Pos &other): bool const
```

objPos class inherits Pos class

```
objPos
+ symbol: char
+ objPos()
+ objPos(xPos:int, yPos:int, sym:char)

+ getSymbolIfPosEqual(refPos:const objPos*): char const
```