# COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

| | |
|---|---|
| Your Team Members | Sirisha Neelamaygum   Mehar Sidhu |
| Team Members Evaluated | hamadeb  yakubup |

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 main program is well organized, it includes comments and is well formatted. Throughout the function objects from different classes such as GameMechs and Player were used through pointers to access the different functions. For example, when generating food they have to call on the GameMechs class. They use the pointer to do this, using the arrow operator to use a function in the class called generateFood(). This happens throughout the function, when they need to use the player class they use the player pointer to access the functions in their Player class. Another good practice we noticed is that they initialize variables that are used often at the top of the function so that it can be reused easily. Lastly, in the LoopDelay function they use an if statement to control the delay for the exit message instead of adding a delay in another part of their function.

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.

C++ OOD approach vs. C procedural design approach in PPA3

| Pros | Cons |
|---|---|
| ● More organized<br>● Faster way to execute the code<br>● Not repetitive<br>● Creates reusable methods and applications with shorter code blocks and development time | ● Difficulty debugging as all classes and files play hand in hand<br>● Testing needs to be done after every implementation to check for errors |

The C approach for PPA3 was easier to use because PPA3 was a smaller-scale project. C code is harder to implement on larger projects because it is a procedural code and it doesn't have structured organized classes. This makes one change cause many other changes to be implemented which if not all changes will need extensive debugging. This is why C++ OOD is a better-organized form of coding for larger projects with many implementations and features.

## 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.

The code does offer sufficient comments as in every file almost every line of code is explained with comments to make it more understandable to the observer. Their comments helped me understand the code blocks and how they implemented the iterations throughout the project. Regarding comments, I would make no additional improvements.

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 function has good indentation, for example in their if statements they indent the actions below the condition, which improves the readability of the code. There is good white space, for example, before a new function, there are a few lines before the next function. In their code they add a newline after something is printed, this would ensure that there is no overlap in the terminal when things are printed out. I did not observe any shortcomings.

## **<u>Peer Code Review: Quick Functional Evaluation</u>**

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 game runs smoothly but there is a small bug. Their exit condition was / but when we started playing the game and then exited their print statement was "you win!" If we didn't start playing the game and exited with the slash it says "forced exit!" This should be changed so that anytime slash is clicked it should say "forced exit!" regardless of if the game has been played or not. In order to debug this they should check their switch case. They should add breakpoints inside the switch case or try to move that condition somewhere else other than the switch case.

Side Note: My partner and I have macbooks so their code did not run on our MacOS but it ran on windows vscode.

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.

The snake game did not cause any memory leaks when using drmemory to check leaks.

**Project Reflection**

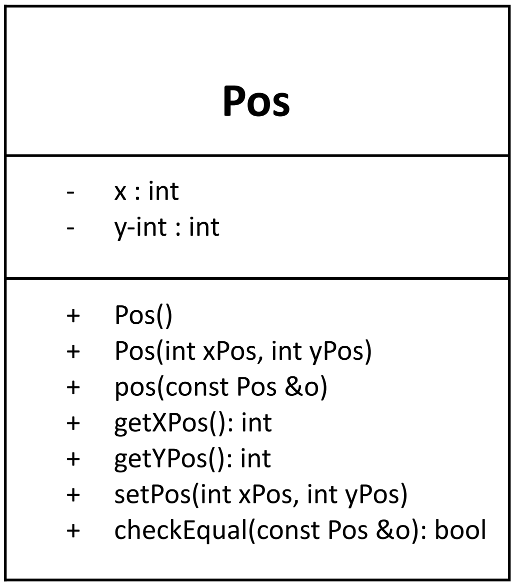
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

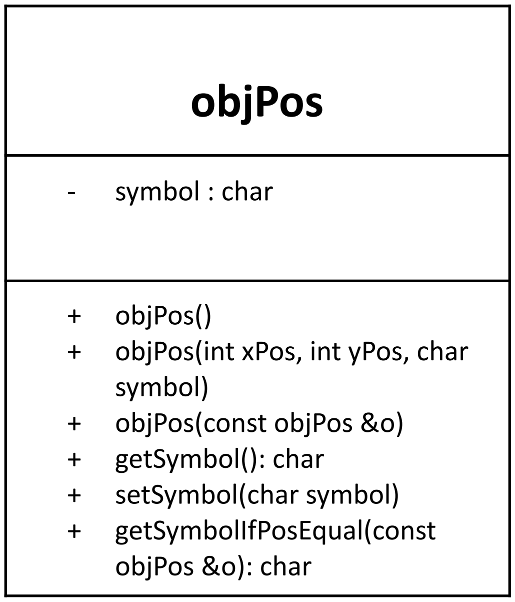
1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

I think the compound objPos class design with the additional pos structs is sensible because this Pos struct allows the code to be more intuitive as it holds the x and y coordinates representing the 2D coordinate point. This was useful when designing with a 2D grid so it could be used in many classes that required the x-y coordinates. By using these coordinates in a struct it was also easier to not duplicate logic for handling coordinates in many places as the struct was easily accessed as pointers in the code implementation.

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(s).

An alternative method to the objPos class design can be adding another class for Pos, this would include a constructor, destructor, copy constructor, getter methods and setter methods. The objPos class can inherit the Pos class. The Pos class would be responsible for all 2D grid design while the objPos is responsible for managing the symbol for the 2D grid. Inheritance makes the program more functional and reusable, for example if we want to make a change to the Pos it doesn't mean that the objPos has to change.


Pos
- x : int
- y-int : int
+ Pos()
+ Pos(int xPos, int yPos)
+ pos(const Pos &o)
+ getXPos(): int
+ getYPos(): int
+ setPos(int xPos, int yPos)
+ checkEqual(const Pos &o): bool


objPos
- symbol : char
+ objPos()
+ objPos(int xPos, int yPos, char symbol)
+ objPos(const objPos &o)
+ getSymbol(): char
+ setSymbol(char symbol)
+ getSymbolIfPosEqual(const objPos &o): char