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

Your Team Members ___abbassj__________ ______mannic5___

Team Members Evaluated (stamout) _____stamout_______ ______shaikh22______

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.

   Objects interact very clearly and concisely here, and the use of comments and spacing helps readability. The printing sequence is well displayed and thorough; however, there is a variable of playerhead that is made but unused.

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
   - Easy modularity and readability
   - Allows for parallel work to be combined as a final result
   - Less global variables that can cause issues.
   - Easier Memory heap access (list will always be destroyed so can just delete the pointer to it)
   - Every Action/item can have all its functions and variables be easily accessible in one place

   Cons

   - Can be confusing if items and functions are not named accordingly
   - Scope can be an issue as some functions may or may not have other functions in scope, if not then a call to the other class needs to be added

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 have sufficient comments with each loop or if statement having a small explanation to follow the code easier. I personally would not change anymore or add anymore I believe it has the exact amount of understanding without going to explain every little detail.

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.

   Yes, it utilizes great indentation and spacing, it does not have issue regarding the readability of the code. I would not change anything about the formatting in this code.

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 game is smooth and has no visible bugs.

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.

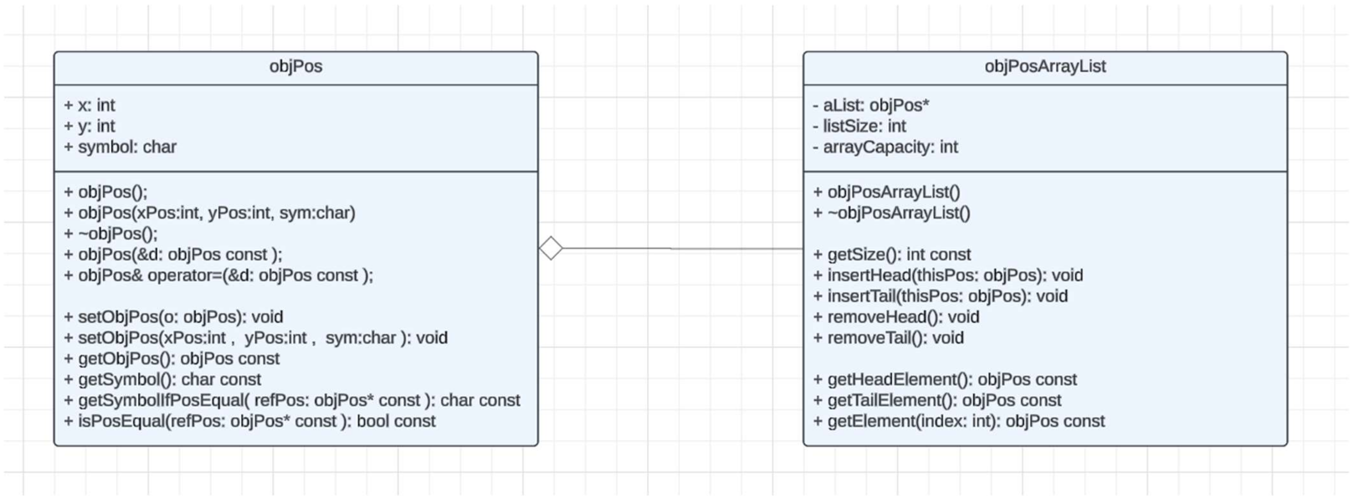
   The game causes a memory leak due to the MacLibUI but not due to any of its own features; However, it causes a huge amount of invalid heap arguments. This is due to attempting to delete the objPos Pos pointer with delete[] but Pos is not an array.

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?
   Not entirely. The compound object design between objPos and objPosArrayList is sensible as differentiating them makes it possible to define an object as both a singular element and as an array. This saves memory as not everything instantiated needs to be an array of positions and an array of just one element is wasteful and confusing. However, having Pos as a struct outside of objPos is not as sensible. As said in the project manual, it can be used to save memory, but, in the case of this project, all objects made with objPos will have a position which discards any memory saving that keeping Pos as a struct may have. Pos as a struct only causes confusion and code complexity in the case of this project.

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).
   In this project, a simpler design where the x and y positions are contained within the objPos class would have the same result as having Pos as a struct. Having the x and y positions contained within the objPos class removes a step when writing the code as now instead of having to write getObjPos()->pos.x to get the x position of the object, it can now be just getObjPos().x. This saves a lot of time and complications throughout the design of the game.


objPos
+ x: int
+ y: int
+ symbol: char
+ objPos();
+ objPos(xPos:int, yPos:int, sym:char)
+ ~objPos();
+ objPos(&d: objPos const);
+ objPos& operator=(&d: objPos const);
+ setObjPos(o: objPos): void
+ setObjPos(xPos:int , yPos:int , sym:char): void
+ getObjPos(): objPos const
+ getSymbol(): char const
+ getSymbolIfPosEqual(refPos: objPos* const): char const
+ isPosEqual(refPos: objPos* const): bool const
objPosArrayList
- aList: objPos*
- listSize: int
- arrayCapacity: int
+ objPosArrayList()
+ ~objPosArrayList()
+ getSize(): int const
+ insertHead(thisPos: objPos): void
+ insertTail(thisPos: objPos): void
+ removeHead(): void
+ removeTail(): void
+ getHeadElement(): objPos const
+ getTailElement(): objPos const
+ getElement(index: int): objPos const