# COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members Dhanvin Tandon, Julian Daneman

Team Members Evaluated mirsab, mohammad

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.

Going through the main program, it was quite easy to interpret, the code was appropriately spaced-out with comments sprinkled throughout to increase readability. The variable and function names clearly indicated their purpose which made the code easier to read. Something to improve on would be removing unneeded temporary comments once the programming is complete.

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.
    a. OOD approach Pros
        i. Much easier to code collaboratively due to modular style of programming
        ii. Easier to implement new features since classes and objects can be used in different parts of the program
        iii. Classes hide implementation details so the code becomes easier to read
    b. OOD approach Cons
        i. Slight performance drop since there is more memory usage in the objects

## **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.

Their main strength was that there are some comments present, especially in areas where logic was changed (e.g., //it3, //finishing touches), and some key parts of the code are explained, such as in the movePlayer() method. On the other hand, there is a lack of detailed explanations. While there are some comments, a lot of them are short or vague (e.g., // re-wrote this section or // moved to keep track easier). More detailed comments would help explain the logic behind the changes. Also, a lot of personal thoughts were left as comments which are quite frankly irrelevant to the commenting process and therefore shouldn't be there. For example, "//^pray this works ima kms if it don't" is not constructive to the commenting process.

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, they had consistent indentation, with each block of code properly indented, making it easy to follow the structure of the code. They also had a clear separation between methods, with each separated by a blank line, making it easy to distinguish between them.

## 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 overall game was mostly bug free. However, at the end the phrase "press enter to exit game" printed a concerning number of times. A possible root cause for this is that the main loop was not paused after printing the message while awaiting user input to exit the game and so a malfunction of the MacUI clear screen method caused a cascading failure.

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.

There were no memory leaks.

## 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?

No. Since there were only three class variables, it was unnecessary to implement a struct; the class would have been very clear and readable without.

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.</u>

I would improve the object design by replacing the pos struct with two simple integer variables for x and y.

| **objPos** |
|---|
| - x: int<br>- y: int<br>- symbol: char |
| + objPos()<br>+ objPos(xPos: int, yPos: int, sym: char)<br>+ setObjPos(o:objPos): void<br>+ setObjPos(xPos:int, yPos:int, sym:char): void<br>+ getObjPos(): objPos const<br>+ getSymbol():: char const<br>+ isPosEqual(refPos:const objPos*): bool const<br>+ getSymbolIfPosEqual(refPos:const objPos*): char const |