CSCI 375 Project #2

Due date: July 9, 2025
How to submit? Email your source code and sample result

In this lab, you will simulate one of the classical synchronization problems to see how the (semi) critical section problem could be implemented using binary and counting semaphores.
The lab is due in 2 weeks, July 11, 2024.

4 readers and 2 writers characterize 6 processes.
Up to three reader processes can be inside their critical section without any writer process. For the writer process to go into its critical section, it should check whether any reader or writer process is in the critical section.

The critical section in this problem is reading the shared data buffer for the reader and updating the shared data buffer for writer processes. It is up to you to implement any shared data for readers and writers, but you must specify the following things in your sample output.

• When the reader or writer enters its critical section, it has to report whether there are any reader(s) or writer(s) other than itself.
• You may print out the data you read or write when implementing an actual buffer. (Optional)
• You must print out "Panic Messages" when the rules behind this semi-critical section problem are not observed.

You run the random number generator function in your main program to choose a process to execute. The selected process starts (resumes) execution, and after one instruction, it will be returned. (You should force each process to run precisely one instruction, then return and wait for its turn.)

You can implement this using a switch statement in C or C++. Do not use any multi-threading or mutex feature from a programming language. *Each process is one big switch statement and will be returned after each instruction. You need to keep track of the program counter of each process to resume at the right place once it is chosen to run by keeping a global counter variable per process.*

Subproject 1: You should implement ***binary and counting semaphores*** as studied in the class for this project.

Subproject 2: You should implement a ***TestandSet*** function as studied in the class for this project.

You should not copy from others or let other students use your code. (I might ask you to explain your code as well.) Violation to this policy will result in automatic fail.