CSCI 375 Project #3

Due date: May 1, 2025
How to submit it? Email your source code (with proper comments) and output with the analysis report

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.

Three readers and two writers characterize six processes.
Up to two 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 reader processes and updating the shared data buffer for writer processes. Implementing real shared data for readers and writers is optional, but you must specify the following things in your sample output.

• When the reader or writer process enters its critical section, it must report whether any reader(s) or writer(s) other than itself.
• You may print out the data you read or write when implementing a buffer. (Optional)
• You must include the "Panic Messages" generating function when the rules behind this semi-critical section problem are not observed. If your solution is correct, the panic message should not be printed.

You run a random number generator function in your main program to choose a process to execute. The selected process starts (resumes) execution. After single instruction, it will be returned to the main function, and the following process to run will be chosen by the result of another random number generator function. (You should force each process to run precisely one instruction, then returns and wait for its turn.)

Do not use multi-threading, mutex, and semaphore features from the programming language and operating system level. You are building these from scratch in this project, not using existing tools including OpenMP. You should implement this project with a simple switch statement in C or C++.

Each (reader or writer) process comprises one big switch statement and will be returned after single instruction. You need to keep track of the global program counter per process to resume at the right place.

Subproject 1: You should implement this with ***binary and counting semaphores***.

Subproject 2: You should implement this with the ***TestAndSet function***.

You should not copy from other students or let other students use your code. Violation of this policy will result in an automatic F grade for both parties.