Assignment 3: Part II Report

SYSC4001: Operating Systems

Group member 1:	Rameen Rahman
Group member 2:	Lindsay Xu
Submitted on:	12/01/2025

Critical Section Problem

Solving the critical section problem involves 3 requirements:

Mutual Exclusion: Only one process can enter a critical section at a time.

Progress: The choice of which process enters the critical section cannot be postponed indefinitely.

Bounded Waiting: A waiting process can't wait indefinitely due to other processes entering the critical section

The majority of these issues are resolved using semaphores, which were used in the code. Semaphores are used for synchronization to prevent race conditions (simultaneous change to a shared variable) between multiple processes. For example, the wait() system call can be used using the associated semaphores. This call ensures that a process can't enter a critical section until there are no other processes in the section. This solved the issue of mutual exclusion. Another system call that can be used is signal(). This system call allows the processes waiting in queue to 'wake up' and use the critical section after the previous process has finished. This helps solve the remaining two issues. For example, the queue follows an algorithm, typically FIFO, meaning that there is no halt in progress, as there is a system to choose the next process. Furthermore, using a system such as FIFO prevents bounded waiting as there is no priority and therefore no indefinite interrupts.

Deadlocks and Livelocks

There are two main instances in which processes cease making progress:

Deadlock: A situation where two processes are each waiting for the other to release a resource, leading to a stall in progress.

Livelock: A situation where two processes make changes in response to each other, but don't make any progress in their tasks.

When testing the code for part 2, no deadlocks were encountered. This made sense as within the program, a TA process would first access an exam, and release it before accessing the rubric. This meant that another TA process could be accessing the exam in the meantime, and wait for the rubric to be released, which is what is expected. As for a livelock, the only situation in which this occurred was when an exam had been graded, but the TA processes were unable to proceed to the next exam, meaning they were continuously confirming that the exam had been graded and making no further progress. This situation should only occur once the last exam has been reached, at which point the program should terminate.