SYSC4001: Assignment 3 Report

Anirudh Sridhar, 101304877

Panshul Bansal, 101294162

Part 2

Part 2.c

I ran both programs from Part 2 (2a and 2b) multiple times and checked the contents of the generated output in the terminal after each run.

Part 2a – Shared Memory without Semaphores

In Part 2a there are no blockings (no semaphores, no sleep inside a loop waiting for another process), so the processes never block each other. Instead, the student and the two TA processes simply run parallely their accesses to the shared memory according to the scheduler.

On my runs there was no deadlock or livelock, all processes finished, and all exams were fully written. However, the execution order of the processes was nondeterministic. Different runs can produce different outputs, for example, one TA may finish marking most exams before the other, and then the TA catches up later. This behaviour comes from the lack of proper synchronization and the presence of race conditions, not from deadlock.

Part 2b – Shared Memory with Semaphores

In Part 2b, semaphores are introduced to coordinate access to the shared memory and the exams. Each step (student producing an exam, TA consuming/marking it, and updating the shared data) is protected by semaphore operations. Every sem_wait has a corresponding sem_post, and the semaphores are always run in the same order, so there is no circular wait condition.

On my runs there was again no deadlock or livelock. The typical execution order is:

1. The student process loads an exam into shared memory and signals via a semaphore that an exam is ready.

2. One of the TA processes wakes up, marks the questions for that exam, updates the shared data, and signals completion.

3. The student moves on to the next exam and repeats the cycle until all exams are processed.

4. Once all exams are done, the terminate flag is set and the TA’s exit cleanly.

Because the semaphores enforce a clear pattern and prevent more than one process from modifying the shared structure at the same time, the system makes progress and terminates without deadlock or livelock.