Report Part C

SYSC 4001A (L3)
Student 1: Eshal Kashif (101297950)
Student 2: Emma Wong (101297761)

GitHub Link: https://github.com/EmmaWong8/SYSC4001_A3_P2

Deadlock and Livelock Discussion

We tested both part2a and part2b several times with 3 TA processes. In all runs, the programs progressed through the full list of 20 exams and eventually loaded the exam for student 9999. After marking all questions for student 9999, each TA printed a "Detected finished" message and the parent process printed "All TAs have finished. Exiting part2x." This confirms that no processes remained blocked or stuck in an infinite loop (therefore, no livelock or deadlock).

For Part 2.a, there are no semaphores at all, so the TAs never block waiting for one another. They may corrupt shared state (for example, multiple TAs sometimes mark the same question on the same exam, or load the next exam at the same time), but they always eventually reach the last exam and terminate. Because there are no blocking synchronization operations, deadlock cannot occur in this version.

For Part 2.b, we introduced semaphores to coordinate access to the shared exam state and to the rubric. A semaphore protects the question state so that each question is only assigned to one TA at a time, and another semaphore serializes rubric corrections and updates to the rubric file. Only one TA loads the next exam at a time. No TA ever holds one semaphore while waiting to acquire another, and every critical section is followed by a signal operation, so there is no circular waiting condition. In practice, all exams are processed in order and the program terminates cleanly on each run, so no deadlock or livelock was observed.

Execution Order Discussion

Although the exact interleaving differs each time (because of random delays and OS scheduling), the overall execution pattern is consistent in both parts. All TAs begin by attaching to the same shared memory and working on the same current exam. For each exam, every TA reviews the five rubric entries. In both parts, this produces several lines such as "Checking rubric for question X," and in both programs multiple TAs can apply corrections to the rubric. In Part 2.a that happens concurrently without protection; in Part 2.b the corrections are made through a semaphore so the updates to the file and shared memory remain consistent.

After the rubric review, the TAs start marking the exam's questions. In Part 2.a, because there is no mutual exclusion, multiple TAs sometimes mark the same question for the same student, and the output shows duplicate “Marking student XXXX question Y” lines. In Part 2.b, the question-selection logic is protected by a semaphore, so each question is only assigned to a single TA, so no duplicates appear.

Once all five questions for an exam are completed, a TA (or the loader logic in Part 2.b) loads the next exam. All TAs then shift to that new student and repeat the same sequence: rubric review followed by question marking. This continues exam by exam until the last exam for student 9999 is processed. At that point, each TA detects that there are no more exams to load and exits.

Overall, the processes run concurrently and interleave their actions, but they always move forward through the exam list in order, finishing each student’s exam before advancing to the next one. The main difference between the two parts is that Part 2.b uses semaphores to coordinate which TA gets to mark which question, whereas Part 2.a leaves those actions unsynchronized, leading to duplicated work.