SYSC 4001 Assignment 3

Part 2c Report

Abdullah Al-Rashid (101310113)

For Part 2a, I ran the program multiple times and it always terminated normally after processing exams 1–19 and then reading the sentinel student 9999 in exam20. There are no semaphores in this version, so there was no deadlock or livelock: all TA processes eventually finished their work and exited. The execution order of the TAs is highly interleaved because it depends on the operating system scheduler. In the output, messages such as "TA k: Checking line i of the rubric." and "TA k: Marking question q for student s." appear in different orders on different runs, but in every case each exam's questions are eventually all marked before the program moves on to the next exam.

For Part 2b, I added two semaphores (rubric_mutex and question_mutex) to protect access to the shared rubric and the completed_questions array. I again ran the program multiple times and it always finished, with the last lines showing that exam 19 was fully marked and exam20 contained the sentinel student 9999, so the program stopped. I also checked the rubric and exam outputs to verify that all questions were marked exactly once per student. This means there was no deadlock or livelock in my implementation. The execution order is now more structured: for each exam, TA 1 is the process that loads the exam file and prints "TA 1: loading exam file …", and then all TAs take turns editing the rubric one at a time because of rubric_mutex. When marking exams, the TAs compete to mark questions, but question_mutex ensures that only one TA at a time selects the next unmarked question, so each question is marked exactly once per student. Because semaphores are always released and are never held while waiting for another semaphore, there is no circular wait and thus no deadlock.