SYSC 4001

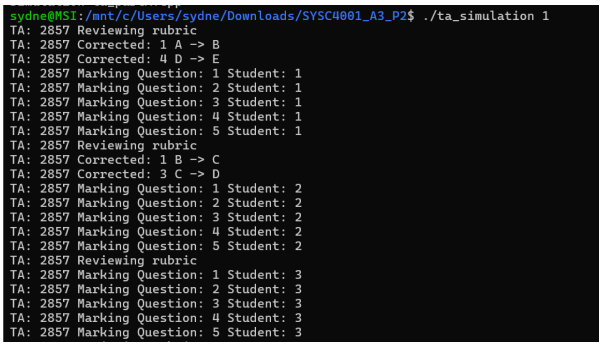
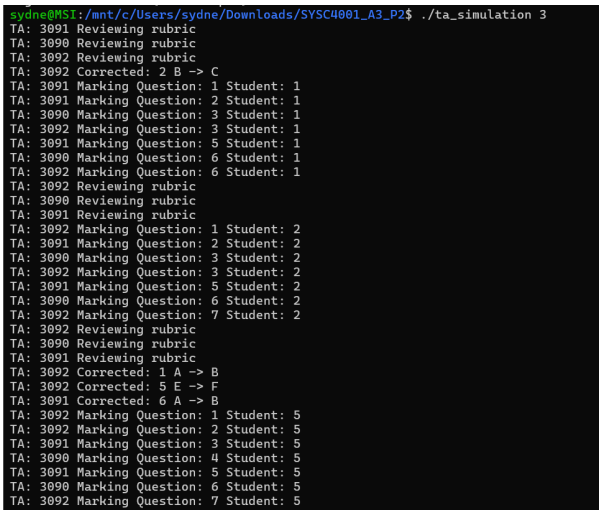
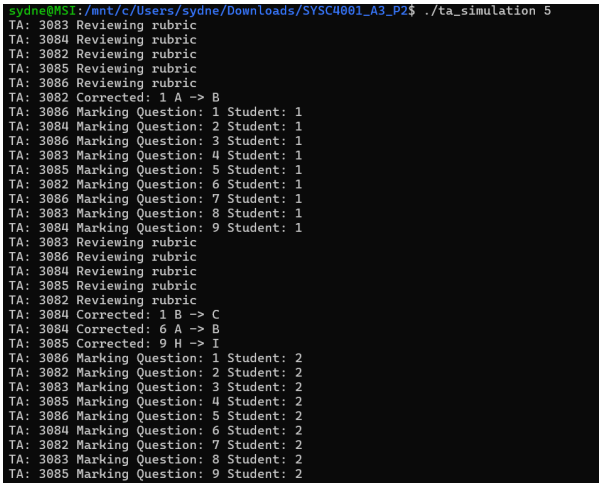
Assignment 3 Final Part 2

Sydney Kuhn - 101302923
(Student 1)

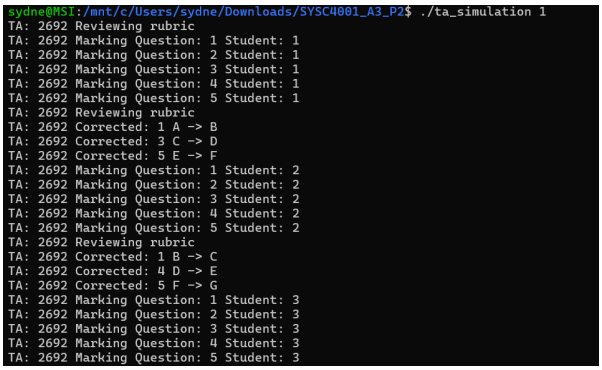
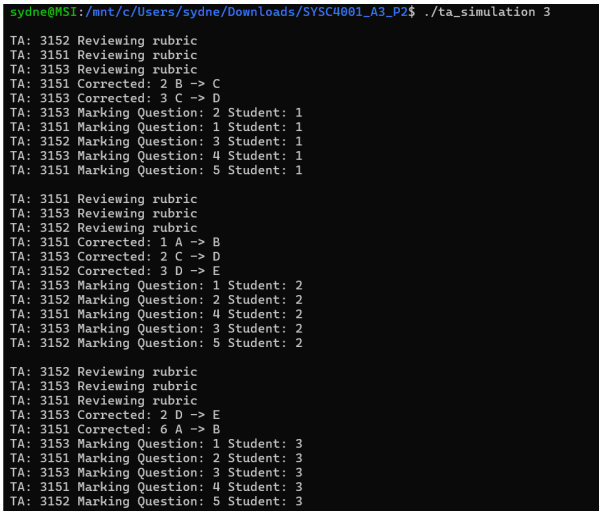
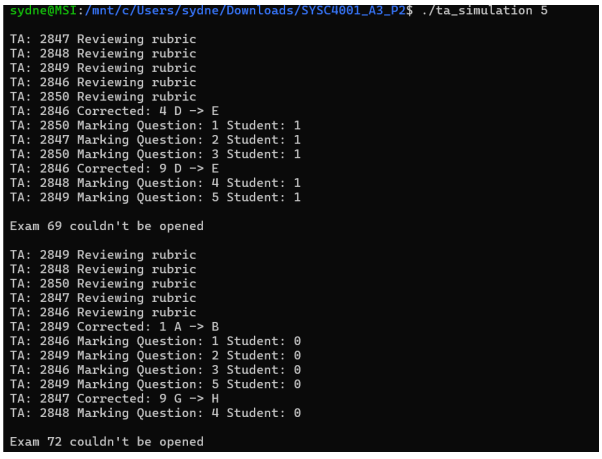
Keon Foster - 101303908
(Student 2)

Repository Link - https://github.com/Jeveloper25/SYSC4001_A3_P2

Part A

	With 1 TA, the program ran smoothly. The TA reviewed the rubric and marked all questions without interference, since no other processes were competing.
	With 3 TAs, some TAs began marking before others finished reviewing the rubric. This caused overlapping outputs and minor inconsistencies. There were 7 questions corrected instead of the intended 5.
	With 5 TAs, race conditions were more obvious. Multiple TAs corrected the same rubric line simultaneously causing 9 questions to be marked. *Note When the program was ran with 8 TA's the program was virtually useless as it was having a difficulty even reading in the rubric and exams.

Part B

	With 1 TA and semaphores, execution was identical to Part A, but controlled access was enforced. No conflicts occurred.
	With 3 TAs, semaphores ensured that only one TA could modify the rubric at a time. Marking proceeded sequentially for each line, avoiding race conditions. There are also only 5 questions marked, as intended.
	With 5 TAs, semaphore control prevented overlapping rubric updates. All TAs marked exams properly and there are still only 5 questions, demonstrating ideal synchronization.

Summary (nxt pg)

In part 3 of this assignment, we experimented to demonstrate the importance of proper synchronization in computer systems. In part A we began to run the TA marking program, without any coordination, this led to all of the processes in a “race condition”. This is when multiple processes are accessing the same data. Another side effect of organized concurrent execution would be either deadlock or livelock. This was clear when we began to see inconsistencies in part A. With just 1 TA the program ran smoothly and as it should, but when more TA’s were introduced the terminal began outputting the wrong values. However when using semaphores in part B, these inconsistencies were not present. A semaphore limits the amount of processes that can interact with the same data at the same time. This is called mutual exclusion, and allows multiple TAs to safely update the rubric and mark exams at the same time. The results of this assignment highlight how proper synchronization can avoid conflicts and maintain consistent access to shared resources.