

## Part C Discussion

In all of my runs of the programs in Part A and Part B, it never entered a deadlock or livelock. To understand why this happened, I looked into why those locks occur, and why my code doesn't enter them. What I found was a deadlock would have occurred if two or more processes stopped because they were waiting for something from each other. This causes them to be stopped infinitely. A livelock would happen if the process kept reacting and changing based on the other process. This causes an infinite loop of basically chasing the tail of the other process. So this means my program always followed a specific safe order of steps that never caused one of these to occur. It shows the code I wrote acts consistent across the many tests.

To address the execution order, the basic sequence followed as this: A process (TA) would get a resource such as a rubric or exam. Then they would hold it and make it inaccessible to the other processes. Then it would do the related processing such as marking or making a correction. If semaphores are not enabled by the command, then the processes will face race conditions that don't block other processes from accessing it. After it lets go of the resource, making it accessible to all other processes. And the cycle repeats until the program encounters the student ID of "9999".

The program cleanly following this structure (with semaphores) allowed no livelocks or deadlocks to occur. Meaning the scheduler never ordered two processes into a dependency cycle. Instead it safely ordered the processes to execute with semaphores where a process needed to wait for a resource to be free to access it to avoid things like read/write errors and execution locks.