After running both part A and part B, and checked the behaviour of both the processes to see if any deadlock or livelock occurred. In all runs, the TA processes were able to review rubrics, sometimes update rubric entries, mark questions, and move on to the next exam. The process continued until the last exam was reached and the finished flag was set. From the logs, you can see that all exams were marked and all TAs exited normally, so there was no deadlock or livelock.

The logs show messages from both TAs, such as reviewing rubrics, changing rubric values, marking questions, and finishing exams. We can also see exams being loaded one after another and the finished flag being triggered. This shows that the processes made real progress, rather than getting stuck waiting for each other or repeatedly interfering.

In Part B, I use two semaphores to protect shared data. sem_rubric ensures only one TA can change the rubric at a time, and sem_exam controls loading the next exam, which enforces mutual exclusion. Each critical section is short and releases the semaphore immediately, so other TAs can continue, ensuring progress. While POSIX semaphores don’t guarantee strict order, each TA waiting for a semaphore will eventually get access, giving bounded waiting in practice. I also protect question selection with a semaphore to prevent two TAs from marking the same question at once. This setup keeps shared data safe and avoids deadlocks or starvation.