

## Experiment Setup

- Number of TAs: 4
- Exam files: “exams/exam\_0001.txt” through “exams/exam\_0020.txt”
- Rubric file: `rubric.txt`

## Observations

### 1. Execution Order

- TA processes start simultaneously.
- Each TA randomly reviews rubric lines and marks a single question at a time.
- After all questions in an exam are marked, one TA loads the next exam.
- Output shows which TA modified the rubric and which TA marked which question.

### 2. Deadlock / Livelock

- No deadlocks or livelocks were observed in any run.
- The use of semaphores ensures mutual exclusion, preventing simultaneous conflicting access.
- The random sleeps and single-question marking per iteration ensure smooth progress without starvation.

### 3. Sample Output Snippet (4 TAs, Exam 1)

Parent loaded exam 1 (student 1)

TA 2 modified rubric line 2

TA 4 modified rubric line 3

TA 2 modified rubric line 4

TA 1 modified rubric line 5

TA 1 marked Question 1 for Student 1

TA 3 marked Question 2 for Student 1

TA 2 marked Question 3 for Student 1

TA 4 marked Question 4 for Student 1

TA 1 modified rubric line 1

TA 1 modified rubric line 3

TA 4 marked Question 5 for Student 1

TA 1 modified rubric line 1

Based on this it is seen that:

- Each TA modifies rubric lines at random
- Each TA marks one question at a time
- The exam\_loaded flag ensures only one TA loads the next exam

### **3. Conclusion**

The program correctly implements mutual exclusion and progress. There were no deadlocks or livelocks observed during the execution of the program. The execution order is non-deterministic due to the multiple TA processes and random timings. This demonstrates concurrency and IPC effectively.