For both part A and part B of part 2, none of the process experienced deadlock or livelock. This is due to the structure of the program. For part A, no semaphores are used, so there is no way for deadlock or livelock to happen. In all the times running the program, there has also been no deadlock or livelock for part B. This is due to the structure of the code and how the semaphores are used. However although part A did not experience livelock or deadlock, it did experience race conditions. This was due to the fact that there was no way for the process to know when data was being used by other processes since semaphores were not used. For example, a race condition that occurred frequently was exam questions being marked twice by different TAs. Both TAs could be accessing the shared data at the same time and do not know about the other process and thus they will both mark a question as marked at the same time. This should not occur as a question should only be marked once. This is why semaphores were added in part B to fix the issues of race conditions in part A. The semaphores in part B work by a TA process acquiring a semaphore before, marking a question, loading an exam, or updating the rubric. At the end of each task, the semaphore is released. There are three different semaphores, one for the exam, one for the questions, and one for the rubric. This use of semaphores solved the race condition issue.