Intro:

In this system we pretty much had multiple processes which were TAs and these processes had to edit a shared file and had access to shared memory with all the other TAs. Each TA first outputs what student number's exam they are working on. Then all the TAs review and maybe even modify the rubric (in our code there is a 50% chance that they actually edit it). After that they go through each question and it the status is unmarked the TA marks the question and this takes between 1 and 2 seconds. Eventually the TAs will reach the last exam which has the student number 9999 and once this exam is fully marked, all the processes stop and finish.

Deadlock and Livelock:

In all our runs there was not really any deadlocks or livelocks that really appeared. The strategy that we used is that we made sure that each process or TA only held one semaphore at a single time. Our rubric semaphore is taken and released right away inside our review rubric function. The questions semaphore is only ever used in short snippets where we have to update a certain question status. The exams semaphore is only ever used to go to the next exam or in a special case where we set the last exam file's student number to 9999 if it isn't already even though it already should be. This all removes the circular waiting condition that causes any sort of deadlocks. Also there aren't any critical sections that loop forever. For example, in the rubric section the TAs are kind of bounded and only iterate through the 5 lines of the rubric. Even in the exam section the TA just iterates through the 5 questions on the exam. As for the livelocks there didn't seem to be any either. This was probably because the logic wasnt exactly complex. The process sees if it can mark a question and if there isn't any available it sleeps and tries again in a bit.