COMP304 Project 2

Collaborators: Ali Oktay (72007) Serkan Berk Bilgiç(71571)

Part I. For the first part we created different queues for each job (land, launch, assembly) and for each pad(padA and padB). To avoid collusion, race condition we used mutexes for each queue. Also we have a count global variable we are using that variable for giving different ids to jobs. When we give id to the jobs or after giving an id incrementing the count because it is global variable and can be destroyed if more than one thread tried to access at the same time we wanted to make this operation atomic. So we added mcount mutex for count variable also. We started count from 1 because we add launching as an initial job and we added it to the launch queue and we gave it id = 0. That's why we started count variable from 1 and incrementing it as we create a new job. So, at time 0 we have a rocket that waits to take off. We are not performing any action without the acknowledgement of control tower. Even for the first take off job we did not add that to the padA queue in main. We are not sure about that so I just wrote that code and commented out in the main. For printing the queue at the given time you should pass "./project_2 -n 10" to be able to see queue printings.

Note: We know both pad A and pad B can be used for the land the pieces but we just used different approach for this. It is not indicated actually in the Project 2 pdf. We give one pad A one pad B. There are different approaches maybe checking the size of the each pad queues' and then giving the job which one has lesser job but we just used different approach. That's not going to affect that much.

Part II. Part I caused starvation because we favored landings to the other jobs to ensure there is no pieces left in the air. To solve this issue, we add maximum wait-time and counter variable for the jobs in the ground. Now, landing pieces still favored but we have an additional conditions. They are not always favored if particular conditions are hold (max wait-time or # of jobs in the waiting queue of other jobs) we will keep waiting landing pieces in the air and favor another jobs for some time until conditions that we are looking for not hold anymore for example no more than 3 waiting assembly or launching jobs. After we did these things now there is a possibility of starvation for launching jobs. Our initial aim was trying to favor landing pieces because we do not wanted them to wait in the air. We still not prefer these we want to allow them as soon as possible starvation of landing pieces is a very big problem so we used similar approaches as given us in a, b options. Aging can be implemented for solving this issue. Actually because of the probability p we might not see starvation of land but when we increase it possibility of starvation for land will increase. We added a counter variable when assembly or launching performed we increment counter and when landing performed we make it 0. We check the value of counter if it is more than the threshold that we decide then we say okay there are a lot of assembly or launching job performed so we will prioritize landing jobs now. That's how we avoid the starvation of landing jobs. There are many other methods to do it but for simplicity we did something like this. Aging is a easy approach to avoid it.

Part III. For this part we added new queue called EmergencyQueue. In discussion board someone asked about it as follows "should we interrupt the ongoing job and continue with the emergency job or just finishing the job in our hand and then allow the emergency ? " and TA answered as "For simplicity let's not interrupt." So we add new mutex for emergency queue operation because we do not want to have a collision. We changed the padA queue if there is already ongoing job we did not

interrupted it but we changed head of the padA and padB queues and added emergency jobs to be able to perform them as soon as possible.

Part IV. For the log purpose we added some more features to the Job struct such as request time and end time. It eases the process when we keep the logs. We used “events.log” file to keep our logs. Also we printed them because it helped us a lot in the process. For printing queues we created a new function and new thread. We check if -n and number given we initialized it to be a -1 and if it remains -1 that means no need for printing else we printed and slept for 1 minutes inside our function.