4a) there are 271 events created for each experiment.

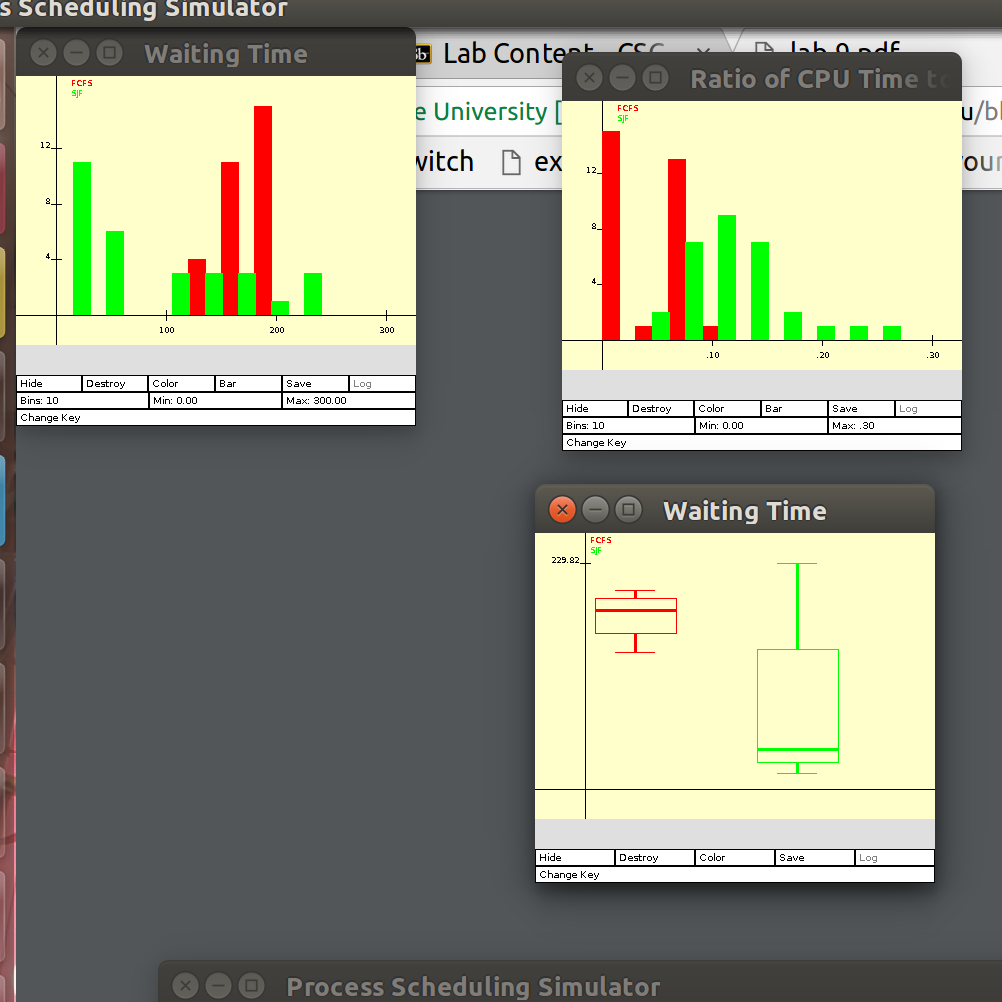
b) the first experiment took 36 milliseconds and the second experiment took 34 milliseconds.

c) 30 processes were created for each experiment.

5c) In this algorithm processes are handled on a first come first served basis, so initially job 30 will have cpu time after everyother job because it is the last job on the list, after the initial run due to the i/o time the order that the jobs are re-entered into the queue becomes different, so afterwards due to job 30’s i/o time it was able to go before certain jobs that have an id number higher than it.

6c) In this experiment since 30 is scheduled in the first half of jobs getting CPU time because its a shorter job than 1-15. It comes after 16-29 because they all are the same size and this causes the algorithm to go in order of id. Due to i/o time after the initial pass job 30 will come back into the queue in a different order than the first pass coming before and after jobs it should come after. And because this is a SJF implementation after a job has CPU time and back in the queue the program checks to see if it is shorter than other jobs and moves it infront of longer jobs.

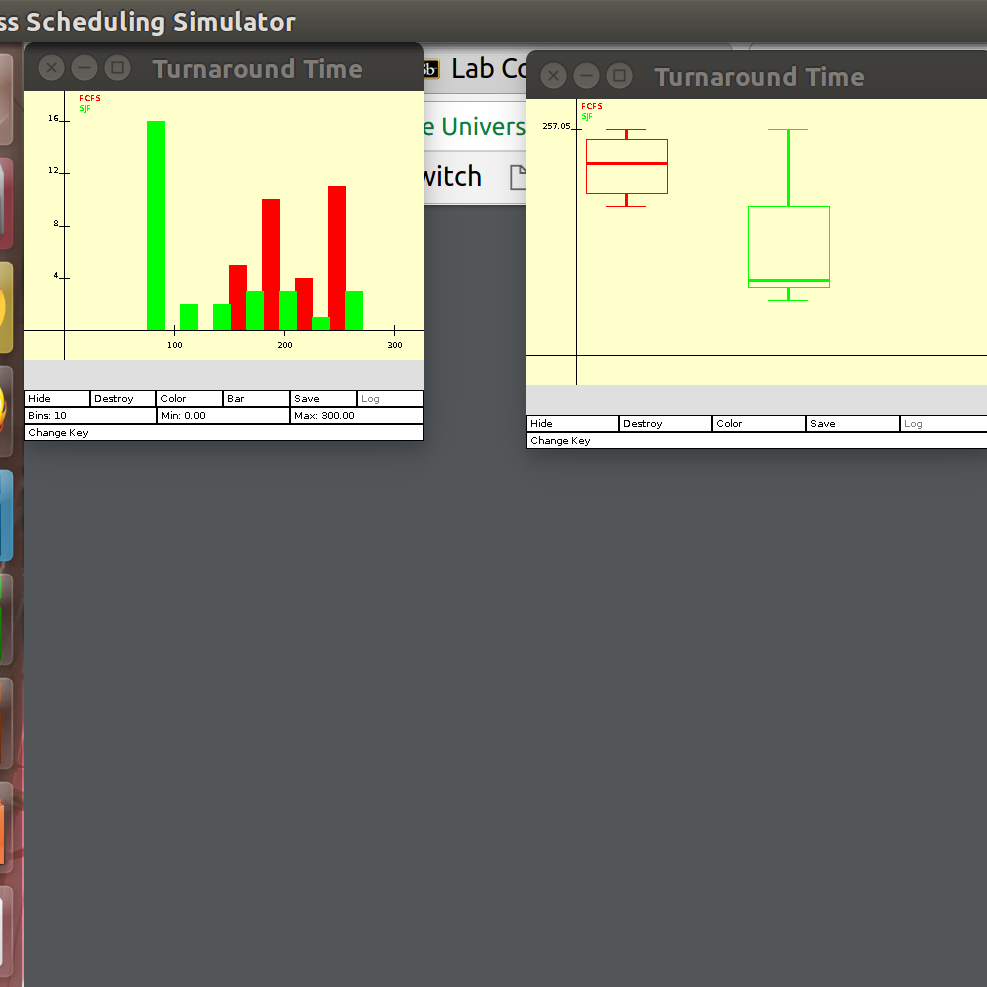
d) Process 30 is scheduled diffterently in both of the experiements due to the algorithms that are used in each. In the first one because it is first come first serve process 30 gets scheduled later than all other processes and then its schedule is determined by the i/o time. In the second experiment it is using a shortest job first algorithm, and since 30 is shorter than many of the other processes it will come earlier and then it will go through i/o time then rescheduled but this algorithm will proritize 30 if it is shorter than other processes.

7a i) The algorithm with the shortest average wating time is the SJF algorithm, looking at the box graph it shows that the algorithm with the lower average is the SJF algorithm.

ii) The wait time is higher in SJF because the process times with the longest cpu burst are waiting for all the processes with shorter times to finish first. This means that the longest process will have to wait until every other process is finished even if the process before it is shorter by a single time unit.

iii) With the FCFS algorithm the wait times may be higher but the wait time of all the processes relative to one another are relatively close, where as in the SJF algorithm the wait times will overall be smaller but the wait time of each process relative to one another is quite distant. For jobs that require all of the processes within a job to finished around the same time regardless of wait times then FCFS is better but if its something like a complex system where multiple process requests are happening and multiple users need their tasks handled then the SJF algorithm may be more efficient to making sure that most of the users are having a smooth experience.

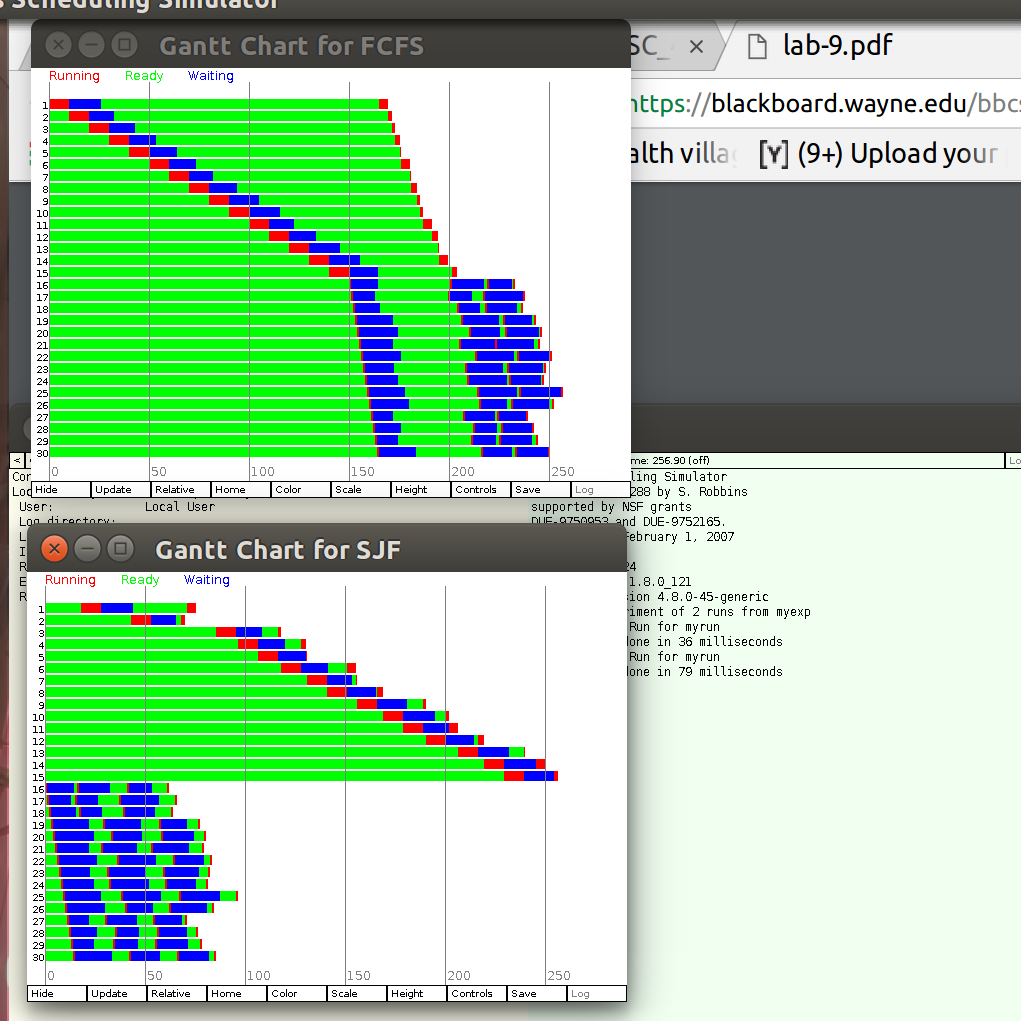
7b i)



Turnaround time is the time that it takes from when a process is submitted for completion and when the complete output is returned to the user.

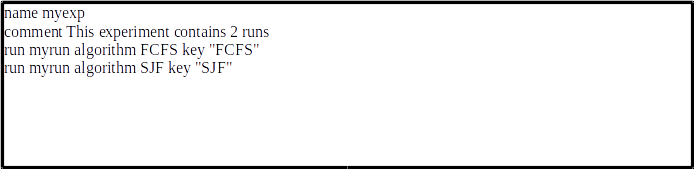
ii) The SJF algorithm has the lower minimum turnaround time, and this is because the SJF algorithm handles the shortest processes first. This means that at the start the processes that need the smallest amount of time to finish go first and then finish keeping the initial turnaround time low, so even if the later processes have larger turnaround time they offset eachother.

c)

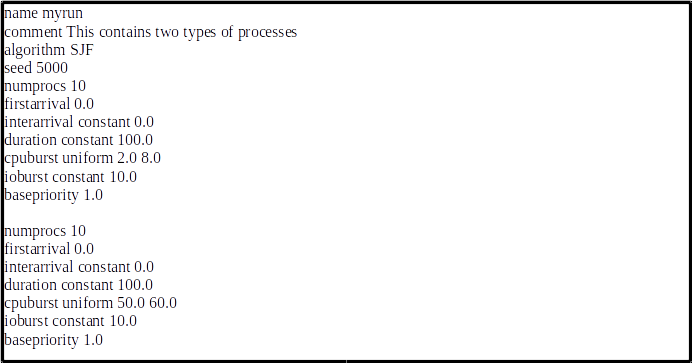


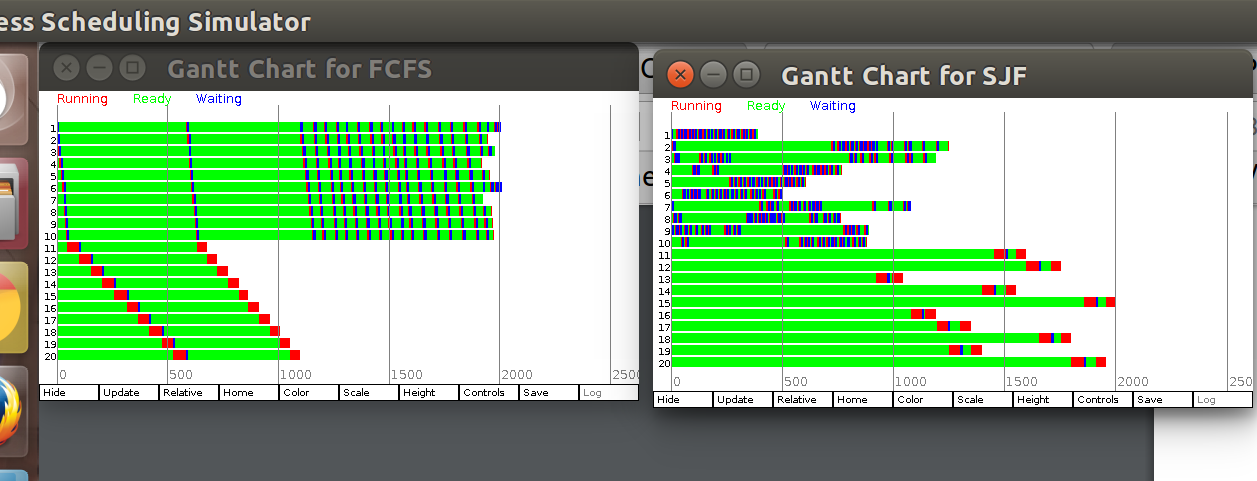
TASK 3

myexp.exp



myrun.run



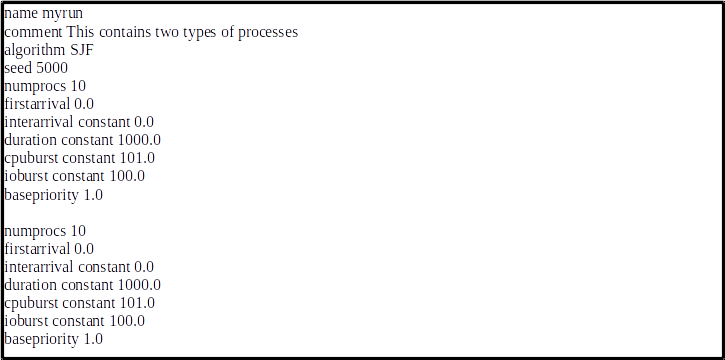


There is a difference between the two figures. In the first come first served chart we see that on the first and even the second pass they follow the processes have cpu time in order but then like earlier i/o time for each process starts to change the order they re-enter the queue so that they terminate in a different order. In the second chart it is following the shortest job first algorithm so the shortest jobs are handled first and we changed it so that the first 10 jobs are the shortest and they get their cpu time and then requeued and since they are shorter still it goes in front of the longer jobs and then the longer jobs get thier cpu time.

TASK 4

myexp.exp



myrun.run



The round robin alogrithm works based on a quantam time and then after each process uses that set amount of time in cpu time it will move to the end of the ready queue. The larger the time quanta the larger the turn around time especially when there is no context based switching. As quantum time increases the round robin algorithm represents the first in first out algorithm, and if the quantum time is too small throughput will suffer with too much overhead.