

## **Project – Scheduling and Deadlock Avoidance**

### **Design approach:**

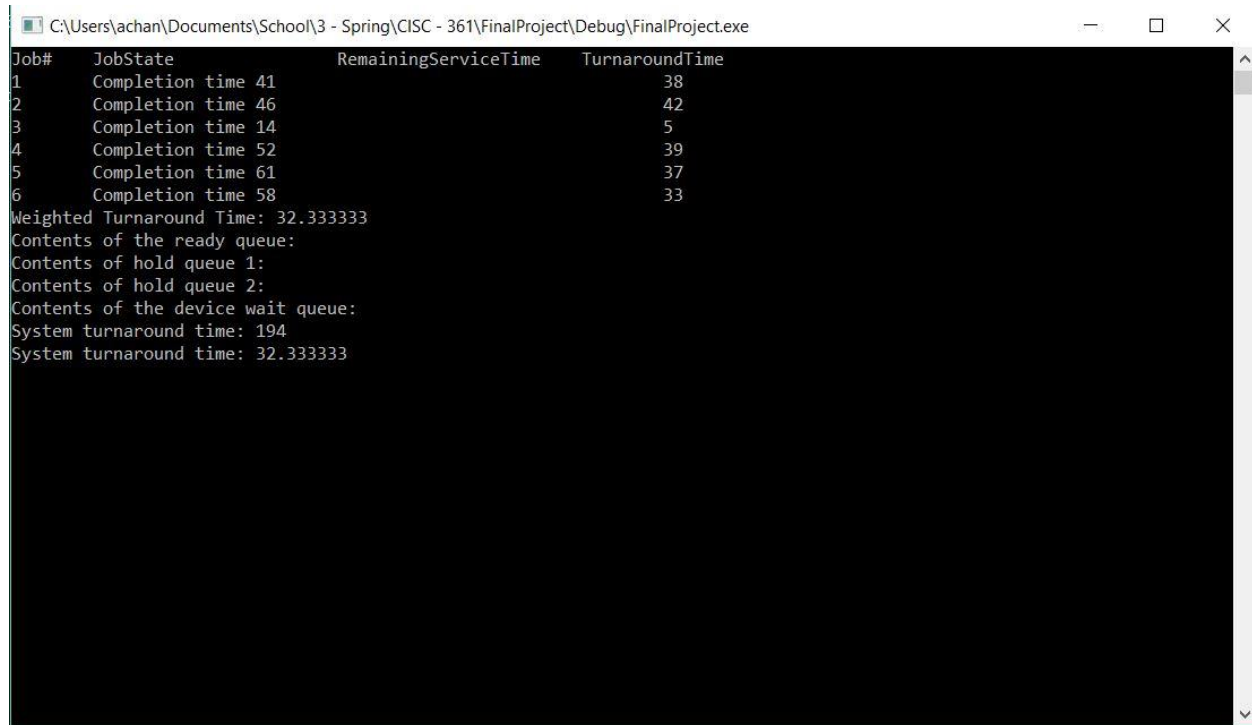
With the design of this lab, we knew that the first design component we would have to implement is a method for our program to determine what kind of instruction was being passed. In order to do this we broke each components into separate if statements. For each if statement it checked if it was C, A, Q, L or D.

When a “C” instruction is read in it is a system configuration command that assigns main memory, serial devices, the simulated start time, and the time quantum/time slice of the system. The “A” instruction is how our jobs arrive, it comes with a job number, priority, time, memory requirements, and devices held. The “Q” instruction is used to request for devices at a certain set time. The “L” instruction is basically the opposite of “Q”, it releases devices at a certain set time. The last instruction was the “D” command, this displays the data in a readable format at a set time/event.

After a new job has left the submit queue it will go to one of two places, either hold queue 1 or hold queue 2. Holding queue 1 was implemented using Shortest Job First (SJF) for its job scheduling. This meant that whatever jobs had the shortest execution time would be selected first from hold queue 1 to be executed; then whenever that job was done, the next shortest instruction would begin. Hold queue 2 was implemented using First In First Out (FIFO) job scheduling. So the first job to enter into hold queue 2 will also be the first job to leave hold queue 2, and so on and so forth with the jobs that follow.

Even in the design portion of this project we knew there was going to be a lot of different variables/jobs that would need to be accessed throughout many different points in the code. Instead of just having a ton of different global variables scattered throughout we implemented a public class called “Job” that held all of the variables and functions we would need later. Some of these variables include the run time, job number, turnaround time, required memory, etc. This made handling each job and the job’s properties much easier later on in the project.

Screenshot of Test Input provided in Lab Instructions:



```
C:\Users\achan\Documents\School\3 - Spring\CISC - 361\FinalProject\Debug\FinalProject.exe
Job#    JobState      RemainingServiceTime  TurnaroundTime
1       Completion time 41                                     38
2       Completion time 46                                     42
3       Completion time 14                                      5
4       Completion time 52                                     39
5       Completion time 61                                     37
6       Completion time 58                                     33
Weighted Turnaround Time: 32.333333
Contents of the ready queue:
Contents of hold queue 1:
Contents of hold queue 2:
Contents of the device wait queue:
System turnaround time: 194
System turnaround time: 32.333333
```