**CSCI 232 - Out Lab 1 (100 pts max)**

**Due: 11pm, September 11, 2020**

**Overview:**

In this lab, you will write a program called JobScheduler; it should take a single input file as a command-line argument. Each line in the input file has the following format:

<job #> <priority> <arrival time (sec)> <duration (sec)>

e.g. a file might contain:

1 3 10 100

5 2 20 50

8 4 5 100

(all values will be positive integers)

These jobs might represent computer programs (or threads) that need to be run by the operating system.

Your task is to write a job scheduler that decides, for each second, what job should be run. The job that is available that has the highest priority value should be scheduled for the next second (ties can be broken arbitrarily). The central data structure to use a priority queue. *You should implement your own priority queue, rather than rely on a built-in data structure.*

**Notes:**

The list of jobs will first need to be sorted by arrival time.

The simulation begins at time 0 and runs until all jobs are finished.

At the beginning of each second:

1. Check if new jobs need to be added to the priority queue.
2. If the queue is non-empty, get the top priority job and run it for 1 second. If it finishes remove it, otherwise put it back in the queue.

**Output:**

Your program should print out which job is running each second. You should also compute the waiting time and total execution time for each job. The waiting time is the time the job actually starts versus when it was first available. The total execution time is the number of seconds that it takes to finish the job once it begins (this will be greater than or equal to its duration). *Some points will be awarded for nicely-formatted output.*

*Example*: suppose a job arrives at time 10s with duration 10s, starts at time 15s and finishes at time 30s. Then the waiting time is 5s, and the execution time is 15s.

Your program should run from the command line:

java JobScheduler input.txt

**Submission:**

Submit using the ecat dropbox prior to the due date/time (you may also submit in-person directly to your TA during lab). Submit your .java files and an input file that you’ve created to test the program (it should have at least 10 jobs) and the resulting output.

*Only one submission per group BUT you must put ALL group member names in the comments of your program.*

**Bonus Problem: Parallel Processing (+ 5 bonus pts)**

Add an additional optional command-line parameter **k** after the filename that indicates the number of processors available to run jobs in parallel. This means that you have k processors available to run jobs, so if multiple jobs are available you can run some jobs simultaneously. Modify your simulator and adjust the output to indicate what job each processor is running at each time. How does adding additional processors affect the job statistics?