The goal of this exercise is to simulate different scheduling algorithms. The algorithms to simulate are:

- First come first served
- Shortest job first
- Preemptive shortest job first
- Round Robin with q=1
- Round Robin with q=5
- Priority based scheduling with a priority scheme that works as follows:
  every job starts with a priority of 0
  every time unit a job is ready but not running its priority is raised by 1
  every time unit a job is running its priority is lowered by 1
  the scheduler should preemptively pick the job with the highest priority to run

Given the following description of a process:

ATL - when the job could first arrive
ATH - when the job could last arrive
TT - how long the process will need to compute
CTL - the minimum time it will run before it does an i/o
CTH - the maximum time it will run before it does an i/o
WTL - the minimum time it will wait for an i/o
WTH - the maximum time it will wait for an i/o

A process can be described as follows:

ATL ATH TT CTL CTH WTL WTH

assume that the distribution of values between the min and the max is flat, so each value is equally likely.

In other words, if ATL==0 and ATH==5, the job will arrive at 0,1,2,3,4,or 5 with equal probability

0 0 16 3 6 4 8 means that the process will arrive at time 0, need to compute for 16 units, it will compute for at least 3 and no more than 6 units between i/o's, and the i/o's will take between 4 and 8 time units. So, the process might behave like this:

compute for 5 seconds and do an i/o; wait for the i/o for 7 seconds
compute for 6 seconds and do an i/o; wait for the i/o for 4 seconds
compute for 3 seconds and do an i/o; wait for the i/o for 5 seconds
compute for 2 seconds and finish

or

compute for 3 seconds and get preempted; wait for 4 seconds; compute of 2 seconds and do an i/o; wait for the i/o for 7 seconds; etc...

0 8 12 1 11 1 1 1 would describe a process that had quick i/o
6 6 7 1 1 0 0 would describe a process that was giving up control frequently but didn't arrive until time 6

we want to simulate the behavior of different job mixes using different scheduling algorithms.

Output:
describe the running of a job as follows: at each time unit, print a 'C' if the job is computing, an 'I' if the job is doing an i/o, an 'R' if the job is ready to run but not running, and an 'X' when the job is done, and a blank if the job has not arrived yet. So, the description for the job above would look like:

CCCCCIIIIIIIICCCCIICCCCIICCX

another job might look like
RRRRRRRRRRRRRRRRRRRRRRRRCCRX

In addition, statistics about the jobs run should include: max finish time, min finish time, turnaround, average wait time, min wait time, max wait time, preemptions, and amount of time with the CPU idle

So, a run might look like this:

FCFS
job1: 0 0 16 3 6 4 8
job2: 5 5 10 4 7 3 3
CCCCCIIIIIIIICCCCIICCCCIICCX finish 33 wait 0 preempt 0
RRRRRRRRRRRRRRRRRRRRRRRRCCRX finish 22 wait 5 preempt 0
max finish 33
min finish 22
turnaround 27.5
max wait 5
min wait 0
average wait 2.5
preemptions 0
idle time 0

Input to the simulation

You will be provided with a file that looks something like this:

2 <- this is the number of jobs to simulate
5 <- this is the number of times to run the simulation for each algorithm
0 0 16 3 6 4 8 <- this describes the first job
5 5 10 4 7 3 3 <- this describes the second job
4 <- this is the number of jobs to simulate
2 <- this is the number of times to run the simulation for each algorithm
0 0 16 3 6 4 8 <- this describes the first job
5 5 10 4 7 3 3 <- this describes the second job
0 0 16 3 6 4 8 <- 3rd
5 5 10 4 7 3 3 <- 4th
... etc

Your simulation should be written in C, C++, or Java. You will need to hand in all source code and output for the program.

extra credit: expand your simulation to support 2 processors and rerun all of the simulations with the two processor model [this probably isn't worth the effort for the amount of credit]