INTERNATIONAL TECHNOLOGICAL UNIVERSITY
2711 N. 1st St., Santa Jose, California 95134
Phone: (888) 488- 4968 https://itu.edu

Project-2 (10 pts)
Instructor: Ahmed Ezzat

Process Scheduling Algorithms

This assignment will give you experience with process scheduling algorithms. Write a Java or C program that performs runs of the following process scheduling algorithms:

- First-come first-served (FCFS) [non-preemptive]
- Shortest job first (SJF) [non-preemptive]
- Shortest remaining time (SRT) [preemptive]
- Round robin (RR) [preemptive]
- Highest priority first (HPF) [both non-preemptive and preemptive]

Run each algorithm for 100 quanta (time slices), labeled 0 through 99. Before each run, generate a set of simulated processes. For each simulated process, randomly generate:

- **An arrival time**: a float value from 0 through 99 (measured in quanta).
- **An expected total run time**: a float value from 0.1 through 10 quanta.
- **A priority**: integer 1, 2, 3, or 4 (1 is highest)

Tip: For debugging, you may want to use the same (pseudo) random numbers each time. For this to happen, you should set the seed of the random number generator to a value, such as 0. For Java, see
http://docs.oracle.com/javase/6/docs/api/java/util/Random.html

Assume only one process queue. Sort the simulated processes so that they enter the queue in arrival time order. Your process scheduler can do process switching only at the start of each time quantum. For this assignment, only consider CPU time for each process (no I/O wait times).

For RR, use a time slice of 1 quantum.

For HPF, use 4 priority queues. Do separate runs for **non-preemptive scheduling** and for **preemptive scheduling** (i.e., consider HPF to be two algorithms, giving 6 algorithms altogether). For preemptive scheduling, use RR with a time slice of 1 quantum for each priority queue. For non-preemptive scheduling, use FCFS.

Each simulation run should last until the completion of the last process, even if it goes beyond 100 quanta. No process should get the CPU for the first time after time quantum 99.

How many simulated processes should you create before each algorithm run? Create enough so that the CPU is never idle for more than 2 consecutive quanta waiting for work to do, except possibly at the very beginning of each run. It's OK to create more processes for a run than you can actually use.

Run each algorithm 5 times to get averages for the statistics below. Before each run, clear the process queue and create a new set of simulated processes.

Outputs for each algorithm run (total 30 runs)

- Each created process's name (such as A, B, C, ...), arrival time, expected run time, and priority.
- A time chart of the 100+ quanta, such as ABCDABCD ...
  Show a process's name in a quantum even if it completed execution before the end of that quantum (so then the CPU is idle at least until the start of the next quantum).
- **Calculated statistics for the processes during the run**:
  - Average turnaround time
  - Average waiting time
  - Average response time
- **Calculated statistic for the algorithm for the run:**
  - Throughput

For each run of HPF, you should compute the averages and throughput separately for each priority queue along with the overall averages and throughput for the run.

Because we're limiting each run not to give the CPU to a process for the first time after quantum 99, your statistics should only include the processes that actually ran. In other words, if you created too many simulated processes for the run, don't count the ones that entered the process queue but never started. This means that

with HPF, if a priority queue has a throughput of 0, starvation probably occurred but you won't know how many processes starved.

Final outputs and report

Final output should be the average statistics over 5 runs for each scheduling algorithm.

In a short report (1 or 2 pages), discuss which algorithm appears to be best for each of the calculated statistics.

Extra credit (Extra 2 points)

Add aging to both HPF non-preemptive and HPF preemptive. After a process has waited for 5 quanta at a priority level, bump it up to the next higher level.

What to turn in

Create a zip file containing:

- Your Java or C source files.
- A text file containing output from your simulation runs.
- Your report as a text file, Word document, or PDF.

Important: Name the zip file after your team and append your section number and the assignment number, for example **SuperCoders-3-2.zip**. Some mailers may not allow you to mail zip files, so you may have to rename the file to have the suffix other than .**zip**, such as .**zzz**. Do not include .**class** files and do not email executable files.

Email your answers to your grader. Your subject line should be:
CSC 502 Assignment #2, Group-n. Cc all your group members so that the grader can do a "Reply all" when needed. This is a group assignment; all group members will receive the same score.