BIRZEIT UNIVERSITY
Electrical and Computer Engineering Department
ENCS339 Operating Systems, First Semester, 2020-2021
Programming Project 1 Instructor: Dr. Adnan H. Yahya**,**
Due: November 10th 2020   Partner Selection Deadline: October 12, 2020

Simulation of CPU Scheduling Algorithms

This assignment is for groups of 2 students each. If you want to do it alone you must get the permission of the instructor through a Ritaj Message. Partner selection should be done by October 12, 2020 (REPLY to a message on Ritaj).

1. Goal

This programming project is to simulate various CPU scheduling policies (3 per group). One of the algorithms should be **Priority** scheduling with **aging**. You need to write a program in a language of your choice (Java recommended) to implement a scheduler (dispatcher) simulator which selects a process from the ready queue for a given scheduling policy and displays scheduling activities. Please, note that this project just **simulates** a CPU scheduler and hence does not require creation of multiple processes or actual execution of tasks. The processes and their arrival times, burst durations and so on are assumed given **in a file**. You may need to use random numbers to generate a stream of jobs with these parameters. When a process is scheduled, the simulator will simply print which task is running at what time, producing Gantt-Chart-like output for the entire process stream. Once all processes are done, you need to generate a table with the performance parameters (TurnAround (TA), WTA, Wait Time, and averages)

2. Specifications

2.1. The scheduling algorithms to be implemented for this project are

1- Round Robin (**RR**) with settable Time Quantum;
2- Mutliprogrammed with uniform I/O percentage.
3- First Come First Served (**FCFS**),
4- Shortest Job First (**SJF**),
5- Shortest Remaining Time First (**SRTF**),
6- Explicit Priority (**EP**) with preemption;
7- Explicit Priority (**EP**) without preemption;

The detailed algorithms are described in the textbook. One difference to note is the default tiebreaker. For all the scheduling algorithms, ties with respect to the main scheduling criteria will be resolved using *PID*; **a lower *PID* will have higher priority for tiebreaking.**

2.2. Task information

The task information will be read from an *input file*. The information for each task will include the following fields.

- *PID*: a unique numeric process ID.
- *arrival time:* the time when the task arrives in the unit of milliseconds
- *burst time (or taskduration):* the CPU time requested by a task
- *repeat:* the number of periods a periodic task will iterate: if 1 it is not repeated.
- *interval:* the arriving interval of a periodic task
- *Deadline:* the max time for program completion

The time unit for *arrival time*, *burst tim*e and *interval* is millisecond, starting from 0. You can assume that all time values are integer and *PID*s provided in an input file are unique. The last two parameters are effective only with periodic tasks for real-time scheduling. For non-periodic tasks, *repeat* will be set to 1 and the *interval* value will be ignored. The main program requires three command-line arguments: *input_file, scheduling_algorithm* and *time_quantum*.
The *time_quantum* is valid only for the RR policy and hence will be ignored for all other scheduling algorithms . The RMS algorithm can only work with periodic tasks while the other algorithms can handle both periodic and nonperiodic jobs**. *FCFS* and *SJF* are non-preemptive, and *RR, SRTF*, EP and *RMS* are preemptive .**

2.2. Task Definition
Each group needs to implement a restricted number of algorithms. The exact details are defined by the of the ID numbers 9last 4 digits) of the team as follows: task ID = ((ID1+ID2) MOD 10). So if ID1 =1987 and ID2 = 2717 then (ID1+ID2) MOD 10 =(1987+2717) MOD 10 = 4704 MOD 10=4.

Task ID	**Policy1**	**Policy 2**	**Policy 3**	**Policy4**
0	**1**	**3**	**5**	**6**
1	**2**	**3**	**4**	**7**
2	**3**	**2**	**5**	**6**
3	**1**	**5**	**7**	**4**
4	**2**	**5**	**4**	**6**
5	**7**	**4**	**1**	**3**
6	**7**	**1**	**3**	**4**
7	**1**	**3**	**5**	**6**
8	**2**	**4**	**7**	**5**
9	**1**	**4**	**5**	**6**

3. Requirements
The basic requirement are to implement the 4 *algorithms* and calculate the *average waiting time*, *turnaround time* and the overall *CPU usage*. You can find the definitions of these metrics in textbook.

4. Evaluations

This project will be evaluated using the following criteria.

- Report: 10%
- Program Structure: 10%
- Documentation and presentation at discussion: 15%
- Robust Execution (No crash): 10%
- Basic Requirements: 55%
- Extra Credits: up to 20%: more algorithms, better interface, using threading

Total: 100 (+20)

5. Submissions

1. **Report**: Write **up to** 3 pages to describe how you designed and implemented your program and list any assumption you made for your project. Describe how to compile and run your program only when special directions are needed and unavoidable. In case you completed some extra credit items, you should describe how to enable and test them. Please, do not repeat in the report the text provided in this description.
2. **Source Code** : Include all the source code you developed or extended from the program. These need to be submitted only electronically (no hardcopies of the code). You will not need more than one program file. The running program needs also to be submitted electronically.

Honor Policy: All are required to adhere to the University honor policy and violations will be dealt with according to University regulations.

Good Luck