Lab 5. Implementing a CPU Scheduling Simulator

(Adapted from http://www.eas.asu.edu/~cse430s4/project/project2.pdf)

1. Goals

This programming project is to simulate a few CPU scheduling policies discussed in the class. You will write a program to implement a simulator with different scheduling algorithms. The simulator selects a task to run from ready queue based on the scheduling algorithm. Since the project intends to **simulate** a CPU scheduler, it does not require any actual process creation or execution. When a task is scheduled, the simulator will simply print out what task is selected to run at a time.

2. Specifications

The selected scheduling algorithms to implement in this project are

1) First Come First Serve (**FCFS**);
2) Round Robin (**RR**);
3) Shortest Job First (**SJF**).

2.1 Task Information

The task information will be read from an input file. The format is

```
pid arrival_time burst_time
```

All of fields are integer type where

`pid` is a unique numeric process ID
`arrival_time` is the time when the task arrives in the unit of milliseconds
`burst_time` is the CPU time requested by a task

2.2 Command-line Usage and Examples

```
Usage: proj2 input_file [FCFS|RR|SJF] [time_quantum]
```

Where `input_file` is the file name with task information. FCFS, RR, and SJF are names of scheduling algorithms. The `time_quantum` only applies to RR.

Examples Description

`proj2 input.1 FCFS` Simulate FCFS scheduling with the data file "`input.1`"
`proj2 input.1 RR 2` Simulate RR scheduling with time quantum `2` milliseconds (4th parameter is required even for quantum 1 millisecond) with the data file "`input.1`"
`proj2 input.1 SJF` Simulate SJF scheduling with the data file "`input.1`"

2.2 Design Hints

No sample code will be given for this project. However, here is a possible design logic you can reference. The simulator first reads task information from input file and stores all data in a data structure. Then it starts simulating one scheduling algorithm in a time-driven manner. At each time unit (or slot), it adds any newly arrived task(s) into the ready queue and calls a specific scheduler algorithm in order to select appropriate task from ready queue. When a task is chosen to run, the simulator prints out a message indicating what process ID is chosen to execute for this time slot. If no task is running (i.e. empty ready queue), it prints out an "idle" message. Before advancing to the next time unit, the simulator should update all necessary changes in task and ready queue status.

3. Requirements

The project requires to simulate FCFS, RR, and SJF scheduling for given tasks and to compute the average waiting time. You can find their definitions in textbook as well as in class slides.

- **Implement scheduling algorithm for FCFS, RR, and SJF.** The program should schedule tasks and print progress whenever the state of any task changes.

- **Print statistical information**. As soon as all tasks are completed, the program should compute and print **average waiting time**.

Note: if you use static array to implement ready queue structure, you can assume the maximum queue length is 20.

5. Submissions

- (70%) Well-commented source code
- (30%) Write a report that
  - describes how to compile and run your program
  - lists at least 3 test cases
  - includes screenshots of your running program for all developed test cases

6. Sample input and output

Input:

The number on the first row is the number of processes, followed by three columns corresponding to *pid*, *arrival_time*, *burst_time* respectively.

4		
0	0	12
1	2	4
2	3	1
3	4	2

Output of Statistical Information (in addition to necessary debugging information):

FCFS:

PID	Arrival Time	Start Time	End Time	Running time	Waiting Time
0	0	0	12	12	0
1	2	12	16	4	10
2	3	16	17	1	13
3	4	17	19	2	13

Average Waiting Time: 9

SJF

PID	Arrival Time	Start Time	End Time	Running time	Waiting Time
0	0	0	12	12	0
2	3	12	13	1	9
3	4	13	15	2	9
1	2	15	19	4	13

Average Waiting Time: 7.75

RR (Time quantum = 5)

PID	Start Time	End Time	Running time
0	0	5	5
1	5	9	4
2	9	10	1
3	10	12	2
0	12	17	5
0	17	19	2

PID	Arrival Time	Running time	End Time	Waiting Time
0	0	12	19	7
1	2	4	9	3
2	3	1	10	6
3	4	2	12	6

Average Waiting Time: 5.5