COEN 346
Programming Assignment #2

Simulating Fair-share Process Scheduling

Problem statement: Implement the simulation of a process scheduler that is responsible for scheduling a given list of processes that belongs to a given list of users. The scheduler is running on a machine with one CPU. The scheduling policy is type of fair-share scheduling and works as follow:

- Several users may exist in the system and each user may own several processes.
- Scheduler works in cyclic manner. This means that in each scheduling cycle it distribute a pre-defined time quantum of CPU between different users and processes. The first scheduler cycle will start at second “1”.
- At any scheduling cycle the scheduler distributes the time quantum equally between different users that have at least one process ready for execution.
- Furthermore, the scheduler distributes each user’s time share equally between processes that belong to that user
- The order which the processes will be scheduled in one cycle is not important, i.e. there is no priority on users and processes.

Example: Assume that at the beginning of a scheduling cycle, users A, B and C exist in the system. User A owns processes P1 and P2 ready for execution. User B owns process P3 ready for execution. User C does not have any ready process. Scheduler should distribute this cycle time quantum 50% to user A and 50% to user B. Therefore, P1 and P2 should receive 25% of quantum usage each, while P3 will receive 50% of quantum. This means that if fair-share scheduler has a quantum equal to 4 seconds. Then in this scheduling cycle P1 and P2 receives 1 second each, while P3 receives 2 seconds of CPU usage.

The input of the algorithms consists of a file named “input.txt” and the output should be written to “output.txt”.

Input:

First line of the input file is time quantum to be distributed by scheduler in each cycle. Next lines contain several blocks corresponding to each user. First line of each bock is the name of the user and the number of its processes. Next lines in each block contain information about processes that belong to the user. For each process “ready time” and “service time” is given. “Ready time” indicates when the process is ready and “service time” is the total CPU usage required by each process. “Ready time” and “service time” values are positive integers, i.e. {1, 2, 3, …}. For example in the following input sample the time quantum for scheduler is 4 seconds (first line). There are two users in the system, A and B. A has two processes, let us call them P0 and P1. P0 will enter the ready queue at the 4th second and need 3 seconds to finish. P1 will enter at the first second and needs 5 seconds to finish. User B has one process which will be ready at the 5th second and needs 6 seconds to finish.

```
Sample “input.txt”:
             4
             A      2
             4      3
             1      5
             B      1
             5      6
```

Output: Set of strings indicating events in the program including:

- **Start and End of each process:** E.g. ***Time: 2, User A, Process 0: Started/Finished***. “Time 2” indicates the occurrence time of the event in seconds. “User A, Process 0” indicates the user and the process related to this event. Moreover “Started” or “Finished” indicates the type of event.
- **Start and End of each time slice:** This event happens when the scheduler pauses or resumes the execution of a process (i.e. load or remove the process from CPU). E.g. ***Time: 15, User B, Process 1: Resumed/Paused***. “Time 15” indicates the occurrence time of the event in seconds, “User B, Process 1” indicates the user and the process related to this event. “Resumed” or “Paused” indicates the type of event.

```
Sample “output.txt” (here we assume that the quantum is 4 seconds):

Time 1, User A, Process 1, Started
Time 1, User A, Process 1, Resumed
Time 5, User A, Process 1, Paused
Time 5, User B, Process 0, Started
Time 5, User B, Process 0, Resumed
Time 7, User B, Process 0, Paused
Time 7, User A, Process 0, Started
Time 7, User A, Process 0, Resumed
Time 8, User A, Process 0, Paused
Time 8, User A, Process 1, Resumed
Time 9, User A, Process 1, Paused
Time 9, User A, Process 1, Finished
Time 9, User A, Process 0, Resumed
Time 11, User A, Process 0, Paused
Time 11, User A, Process 0, Finished
Time 11, User B, Process 0, Resumed
Time 13, User B, Process 0, Paused
Time 13, User B, Process 0, Resumed
Time 15, User B, Process 0, Paused
Time 15, User B, Process 0, Finished
```

Implementation Requirements:

The processes (both scheduler and user processes) MUST be simulated using threads. The scheduler should be able to pause and resume each process and only one process should be running at a time (because the simulated system has only one CPU). This means that the scheduler thread will ask a simulated process (T1) to suspend its execution and gives the CPU to the next simulated process (e.g. T2) based on the discussed scheduling policy. Each process is responsible to keep track of its total execution and inform the scheduler thread when it is finished. This process continues until all processes are done.

The program must work with arbitrary number of threads and arbitrary number of users.

The assignment should be done in a group of three students. The deliverable consists of a well-commented code and a report document. Do not include your code in your report.
The report should be at least two pages with the following sections:

- Name of group members
- High level description of the code (description of the methods/functions/threads/data structures and the flow of the program).
- A detailed conclusion, discussing you experience with using thread for simulating fair-share scheduling (e. g. is the simulation accurate?). You also need to analyze your program and discuss its drawbacks and advantages compared to round robin technique among all the processes independently of the users.

This assignment will take 35% of the total mark for programming assignments. Also, for this assignment 80% of the mark is dedicated to your code and 20% to your report.

The code and the report should be submitted in one zip file through Moodle by Friday March 12th, 2021 (11:59 pm).

Also, you will have to demonstrate your work as a team to your TA during a lab session. This will be done through Zoom.