Topic 1: SIMULATION: CPU SCHEDULING ALGORITHMS COMPARISON

Overview:

In this project, you'll implement one of the following CPU scheduling algorithms by writing a CPU simulator.

1. Shortest Job First (SJF)(Non-preemptive)
   The shortest job first algorithm is a priority based scheduling algorithm that associates with each process the length of the process's next CPU burst. When CPU is available, it is assigned to the process that has the smallest next CPU burst.
2. Shortest Remaining Time Next (SRTN)(Preemptive)
   The shortest remaining process next scheduling algorithm is the preemptive Shortest Job First algorithm. With this scheduling algorithm, the process in the ready queue with the shortest execution time is chosen to execute. If a "new process" arrives in the ready queue with a CPU service time less than the remaining time of the current process, preempt.
3. First Come First Serve (FCFS)
   The first come first serve algorithm is simplest CPU-scheduling algorithm. In this algorithm, the process at the head of the queue is allowed to execute until it voluntarily leaves the CPU due to either termination or an I/O request. In other words, the first come first serve algorithm is a non-preemptive CPU scheduling algorithm.
4. Roundrobin (RR)
   The roundrobin algorithm is similar to FCFS scheduling algorithm, but preemption is added to switch between processes. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of 10, 50, and 100 time quantum for this project.

Your Task

Your simulation structure should be event-driven simulation, which is the most common simulation model. At any given time, the simulation is in a single state. The simulation state can ONLY change at event times, where an event is defined as an occurrence that MAY change the state of the system. Events in this CPU simulation are the following: process arrival and the transition of a process state (e.g., when an interrupt occurs due to a time slice, the process moves from running state to ready state).

Each event occurs at a specified time. Since the simulation state only changes at an event, the clock can be advanced to the next most recently scheduled event. (Thus the term *next event simulation model*.)

Events are scheduled via an event queue. The event queue is a sorted queue which contains "future" events; the queue is sorted by the time of these "future" events. The event queue is initialized to contain the arrival of the processes and the first occurrence of the timer interrupt. The main loop of the simulation consists of processing the next event, perhaps adding more future events in the queue as a result, advancing the clock, and so on until all processes terminate.

Input Format

The simulator input includes the number of processes that enters the system, for each process, the arrival time and the cpu time it needs, the finishing time,

You should assume the processes listed in the input file will be in the order that the processes arrive. All these simulation parameters are integers. Your simulator obtains these parameters from the input file, which is in the following format.

```
number_of_processes
process_number arrival_time  cpu_time
process_number arrival_time  cpu_time
.
.
```

The input file will be provided for the CPU simulation.

Output Format

```
Shortest Job Next (non-preemptive):

Total Time required is 269 time units
Average waiting time is 156 time units


Process 1:
Service time = 59
Turnaround time = 59
Process 2:
Service time = 78
Turnaround time = 297
Process 3:
Service time = 12
Turnaround time = 34
```