Processor Execution Simulator

Assignment Description

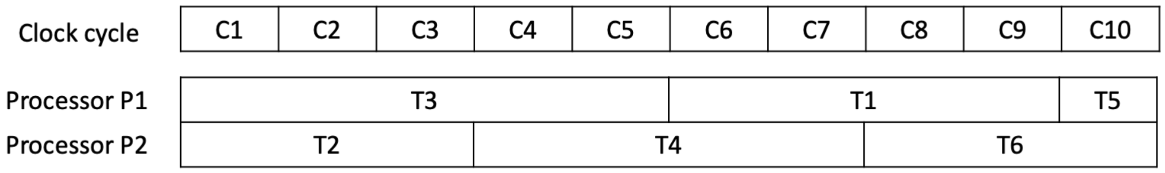
In this assignment, you are required to build a **simulator** that simulates processor execution for processes (i.e., tasks). Below is a description of the simulator:

- The number of processors is fixed during the whole simulation.
- All processors are synchronized, i.e., they all have the same clock.
- Processors are given unique ids as follows: P_1, P_2, P_3, P_4, etc.
- Clock cycles are given unique ids as follows: C_1, C_2, C_3, C_4, etc.
- Tasks are given unique ids as follows: T_1, T_2, T_3, T_4, etc.
- A processor can only execute one task at a time. For example, if we have four processors, then a maximum of four tasks can run in parallel.
- The simulator receives an input text file that describes all tasks' information, where each task is described by three integer arguments:
  - Creation time: the clock cycle at which this task is created
  - Execution time: the number of clock cycles needed to complete the task
  - Priority: which can be either high (1) or low (0).
- The simulator has a queue for holding tasks. In other words, when a task is created, it must be automatically put in a queue.
- The simulator has a scheduler that is responsible for determining the assignment of tasks that are waiting in the queue to processors that are available.
- If there is a processor that is available, then the schedular must prioritize the assignment of tasks with "high priority" over tasks with "low priority".
- If there is a tie, i.e., a processor became available and there are two "high priority" tasks waiting in the queue, then choose the task with the longest execution time. If is still a tie, then choose any task at random. Use the same strategy for breaking ties with "low priority" tasks.
- Task execution cannot be interrupted. In other words, if a task has started executing on a processor, then it holds this processor till its execution is complete.
- Task creation time is negligible. For example, if a task was created during cycle N, and there was an available processor, then this task can be immediately assigned to this processor in the same cycle N.
- As input, the simulator takes three arguments:
  - The first argument is an integer, which represents the number of processors.
  - The second argument is an integer, which represents the total number of clock cycles in the simulation.
  - The third argument is string, which represents the path to the text file that contains tasks' information
- As an output, the simulator produces a cycle-by-cycle report about important events during simulation, such as created tasks, running tasks, completed tasks, and processor's state. I will leave it to you (the student) to determine the exact nature of the report. However, please note that I prefer a simple, readable report over reports that are riddled with so many details. Moreover, I prefer if the report is printed on the console, i.e., do not store the report in files.

Below is an example of an input text file that describes six tasks.

6	The first line describes the number of tasks
1 4 0	Creation time, execution time, and priority of task T1
1 3 1	Creation time, execution time, and priority of task T2
1 5 1	Creation time, execution time, and priority of task T3
3 4 1	Creation time, execution time, and priority of task T4
4 1 0	Creation time, execution time, and priority of task T5
5 3 0	Creation time, execution time, and priority of task T6

Assuming two processors and 10-cycle simulation, the below diagram describes one possible simulation output for the above input task file.



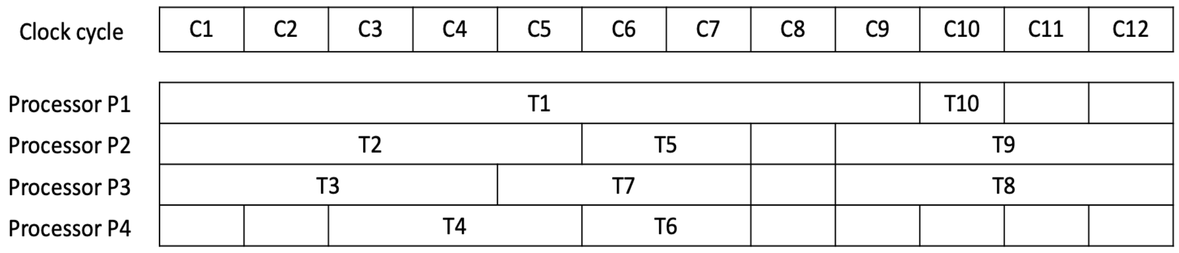
Let us now analyze the simulation output, cycle-by-cycle.

- At cycle 1, tasks T1, T2, and T3 are created and added to the queue, then the schedular dequeues T2 and T3 and assigns them to processor P1 and P2, respectively. Note that the scheduler prioritized T3 and T2 over T1 because they have high priority. Also, note that it would have been also okay if T3 was assigned to P2 and T2 was assigned to P2, i.e., a task can be assigned to any processor.
- At cycle 2, no tasks were created and T3 and T2 are still using the processors. T1 is still waiting in the queue.
- At cycle 3, T4 is created and added to the queue. Therefore, waiting tasks are T1 and T4.
- At cycle 4, T5 is created and added to the queue. Therefore, waiting tasks are T1, T4 and T5. Moreover, T2 has finished execution and therefore processor P2 becomes available. The scheduler assigns T4 to P2 because it has the highest priority.
- At cycle 5, T6 is created and added to the queue. Therefore, waiting tasks are T1, T5 and T6.
- At cycle 6, T3 finished its execution and processor P1 becomes available. The scheduler assigns T1 to P1 because all waiting tasks have low priority but T1 has the longest execution time.
- At cycle 7, T1 and T4 are still using the processors, while T5 and T6 are waiting in the queue.
- At cycle 8, T4 finished its execution and processor P2 becomes available. The scheduler assigns T6 to P2 because all waiting tasks have low priority but T6 has the longest execution time.
- At cycle 9, T1 and T6 are using the processors, while T5 is waiting in the queue.
- At cycle 10, T1 finished its execution and processor P1 becomes available. The scheduler assigns T5 to P1 because it is the only task in the queue.

Here is another example of an input text file that describes 10 tasks.

```
10
1 9 1
1 5 0
1 4 0
3 3 1
4 2 0
4 2 0
5 3 1
9 4 0
9 4 1
10 1 0
```

Assuming four processors and 12-cycle simulation, the below diagram describes one possible simulation output for the above input task file.



I will leave it to you (the student) to do the cycle-by-cycle analysis of the simulation output.

Implementation considerations

- Your code must contain, **at least**, the following classes: Main, Simulator, Clock, Schedular, Task, and Processor.
- Implement the simulator using the principles of object-oriented programming and take into consideration that your simulator is maybe extended in the future.
- To slow things down to a human-level, make the clock cycle time equal to one second.
- Note that at the beginning of a new clock cycle, new created tasks are added to the queue first, **then** the scheduler assigns tasks to idle processors, if any.

Deliverables

1. Source code.
2. An organized report (in pdf format) that contains a high-level explanation of your algorithms and your software design (with a UML diagram).
3. A URL of an unlisted youtube video that contains a **demonstration** of your simulator. You do not need to describe your classes or algorithms in the video. Preferred video time is 5 minutes. At the beginning of the video, make sure to introduce yourself, and state your education and experience. Also, please show your face during the whole video.