CPU Multi-level Scheduling

# **Summary:**

AG Scheduling is a multi-level scheduling that consists of 3 queues, the first queue is **FCFS**, the second queue is **Non-Preemptive Priority**, the third queue is **Preemptive SJF**. **Every process** has name, burst time, arrival time, priority, quantum time. **Note:** The quantum time is different for every process and it is not global as it the case for Round Robin).

The project is built using Java, Intellij.

# **Description:**

Any process arrives for the **first time** should enter the **first level queue** (**FCFS**).

Assume process **P1** arrives with quantum time = 7. It will enter the **first level queue** then the execution will begin, and it will enter the CPU. Since it’s **coming from the first level queue**; it **will be processed for ceil (25%) of the current process quantum** which is **ceil (7/4) = 2-time unit**. If the process still needs more time for processing, it will be moved to the **second level queue** **end** and the **quantum time will be increased for this process by 2** (**quantum time = 7 + 2 = 9**).  
At this moment, The **CPU scheduler** should check if the first level queue **contains any process or** **not**. **If** there’s is any **process exists**, it will be picked and then repeat the steps described above. **Otherwise** the scheduler will pick a process from the second level queue which is Non-Preemptive Priority (**the lowest priority value process will be picked firs**t) and it will be **processed for ceil (25%)** of its current quantum time. **If the process still has burst time**, it will be **moved to the third level queue** and the **quantum** will be **increased by half of current quantum** (**quantum += quantum/2**).

The CPU scheduler should check **if level one queue** **or level two queue has any process waiting**.

# **If** there’s any, then **the previous steps will be repeated** based on which queue the process is coming from. **Otherwise** the scheduler will select a process from **the** **third queue** (Preemptive SJF) with the least remaining burst time. **During execution** for this process; a new process may arrive for the first time and it will enter level one queue. **At this moment** the **execution of the current process** should be **paused**, and it will be **added** **again** to level three queue and the q**uantum time will be increased by the remaining quantum time** (**quantum += remaining quantum**). The scheduler will pick the newly entered process at the first level queue and so on.

# **Overall:** The scheduler should be able to pick a process from the third queue if and only if the first and second queues are empty. Also, the scheduler should be able to pick a process from the second queue if and only if the first level queue is empty.