Assignment # 3
Operating system I

CPU Schedulers Simulator

Write a java program tosimulate thefollowing schedulers:

1. **preemptive** Shortest-Job First (SJF) Scheduling with context switching
2. Round Robin (RR) with context switching
3. **preemptive** Priority Scheduling (with the solving of starvation problem)
4. AG Scheduling :

a. Each process is provided a static time to execute called quantum.

b. Once a process is executed for given time period, it's called **FCFS till** the finishing of (ceil(25%))of its Quantum time then it's convertedto**non-preemptive Priority till** the finishing of the next (ceil(25%)),after that it's converted to**preemptive** Shortest- Job First (SJF) .

c. We have 3 scenarios of the running process

i. The running process used all its quantum time and it still have job to do (add this process to the end of the **queue**, then increases its Quantum time by **Two**).

ii. The running process was execute as **non-preemptive Priority** and didn't use all its quantum time based on another process converted from ready to running (add this process to the end of the **queue,** and then increase its Quantum time by ceil(**the remaining Quantum time/2**)).

iii. The running process was execute as **preemptive** Shortest- Job First (SJF) and didn't use all its quantum time based on another process converted from ready to running (add this process to the end of the **queue,** and then increase its Quantum time by **the remaining Quantum time**).

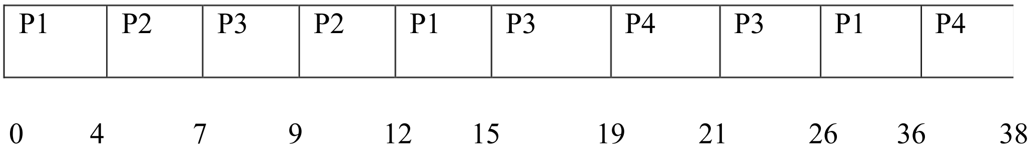
iv. The running process didn't use all of its quantum time because it's no longer need that time and the job was completed (set it's quantum time to **zero**).

Example :

Processes	Burst time	Arrival time	Priority	Quantum
P1	17	0	4	7
P2	6	2	7	9
P3	11	5	3	4
P4	4	15	6	6

Answer:

- Quantum (7, 9, 4,6) -> ceil(25%) = (2,-,-,-) && ceil(50%) = (4,-,-,-)
- Quantum (7+3,9,4,6) -> ceil(25%) = (-,3,-,-) && ceil(50%) = (-,5,-,-)
- Quantum (10,9+3,4 ,6) -> ceil(25%) = (-,-,1,-) && ceil(50%) = (-,-,2,-)
- Quantum (10,12,4+2,6) -> ceil(25%) = (-,3,-,-) && ceil(50%) = (-,6,-,-)
- Quantum (10,0,6,6) -> ceil(25%) = (3,-,-,-) && ceil(50%) = (5,-,-,-)
- Quantum (10+4,0,6,6) -> ceil(25%) = (-,-,2,-) && ceil(50%) = (-,-,4,-)
- Quantum (14,0,6+2,6) -> ceil(25%) = (-,-,-,2) && ceil(50%) = (-,-,-,3)
- Quantum (14,0,8,6+2) -> ceil(25%) = (-,-,2,-) && ceil(50%) = (-,-,4,-) && (-,-,5,-)
- Quantum (14,0,0,8) -> ceil(25%) = (4,-,-,-) && ceil(50%) = (8,-,-,-) && (10,-,-,-)
- Quantum (0,0,0,8) -> ceil(25%) = (0,0,0,2) && ceil(50%) = (-,-,-,4)
- Quantum (0,0,0,0)



Program Input

Number of processes
Round robin Time Quantum
Context switching

For Each Process you need to receive the following parameters from the user:

Process Name
Process Arrival Time
Process Burst Time
Process Priority

Program Output

For each scheduler output the following:

Processes execution order
Waiting Time for each process
Turnaround Time for each process
Average Waiting Time
Average Turnaround Time
Print all history update of quantum time for each process (**AG Scheduling**)

x You have to implement unit-tests for your code that will use the attached test-cases provided with the assignment.

x The assignment is submitted in group of max. 5 students and min. 4 students.

x Late submission is not allowed

Grading Criteria

BOUNS (10 grades)

	preemptive Shortest- Job First (SJF) Scheduling	Round Robin (RR) Scheduling	Priority Scheduling	AG Scheduling	Grade
Processes execution order Waiting	6	6	6	13	**31**
Time for each process	6	6	6	13	**31**
Turnaround Time for each process	2	2	2	4	**10**
Average Waiting Time	2	2	2	4	**10**
Average Turnaround Time	2	2	2	4	**10**
Print all history update of quantum time for each process (**AG Scheduling**)	0	0	0	8	**8**
Unit Tests	6	6	6	7	25
Grade					**125**