Operating system I
Assignment # 3

CPU Schedulers Simulator

Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system.

Write a java program to simulate the following schedulers:

1. **Non-Preemptive** Shortest- Job First (SJF) (using context switching)
2. Shortest- Remaining Time First (SRTF) Scheduling (with the solving of starvation problem using any way can be executed correctly)
3. **Non-preemptive** Priority Scheduling (with the solving of starvation problem using any way can be executed correctly)
4. AG Scheduling :
    a. The Round Robin (RR) CPU scheduling algorithm is a fair scheduling algorithm that gives equal time quantum to all processes **So All processes are provided a static time to execute called quantum**.
    b. A new factor is suggested to attach with each submitted process in our AG scheduling algorithm. This factor sums the effects of all three basic factors ((random_function(0,20) or 10 or priority), arrival time and burst time)The equation summarizes this relation is:

AG-Factor = (Priority or 10 or (random_function (0,20)) + Arrival Time + Burst Time

    c. A new Random function (RF) is suggested between (0,20) and attached with each submitted process in our AG scheduling algorithm. This RF can update the **AG-Factor** based on the random number.

- If(RF()<10)-> **AG-Factor = RF() + Arrival Time + Burst Time**
- If(RF()>10)-> **AG-Factor = 10 + Arrival Time + Burst Time**
- If(RF()=10)-> **AG-Factor = Priority + Arrival Time + Burst Time**

d. Once a process is executed for given time period, it's called **Non-preemptive AG** till the finishing of (ceil (50%)) of its Quantum time, after that it's converted to **preemptive AG**

- **preemptive AG :** processes will always run until they complete or a new process is added that requires a smaller **AG-Factor**

e. 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 (ceil(10% of the (**mean of** Quantum)))).

ii. The running process 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 unused Quantum time of this process**).

iii. The running process finished its job (set its quantum time to **zero and remove it from ready queue and add it to the die list**).

Example of AG Schedule:

Processes	Burst time	Arrival time	Priority	Quantum
P1	17	0	4	4
P2	6	3	9	4
P3	10	4	3	4
P4	4	29	8	4

Answer:

Processes	Burst time	Arrival time	Priority	Quantum	**Random Function**	**AG-Factor**
P1	17	0	4	4	**3**	**20**
P2	6	3	9	4	**8**	**17**
P3	10	4	2	4	**10**	**16**
P4	4	29	8	4	**12**	**43**

- Quantum (4, 4, 4,4) -> ceil(50%) = (2,2,2,2) P1 Running
- Quantum (4+1,4,4,4) -> ceil(50%) = (3,2,2,2) P2 Running
- Quantum (5,4+2,4 ,4) -> ceil(50%) = (3,3,2,2) P3 Running
- Quantum (5,6,4+1,4) -> ceil(50%) = (3,3,3,2) P1 Running
- Quantum (5+2,6,5,4) -> ceil(50%) = (4,3,3,2) P3 Running
- Quantum (7,6,5+1,4) -> ceil(50%) = (4,3,3,2) P2 Running
- Quantum (7,6+3,6,4) -> ceil(50%) = (4,5,3,2) P3 Running
- Quantum (7,9,0,4) -> ceil(50%) = (4,5,0,2) P1 Running
- Quantum (7+3,9,0,4) -> ceil(50%) = (5,5,0,2) P2 Running
- Quantum (10,0,0,4) -> ceil(50%) = (5,0,0,2) P1 Running
- Quantum (0,0,0,4) -> ceil(50%) = (0,0,0,2) P4 Running
- Quantum (0,0,0,0)

P1	P2	P3	P1	P3	P2	P3	P1	P2	P1	P4

0 3 5 9 12 17 20 21 25 26 33 37

Program Input

- Number of processes
- Round Robin Time Quantum
- context switching

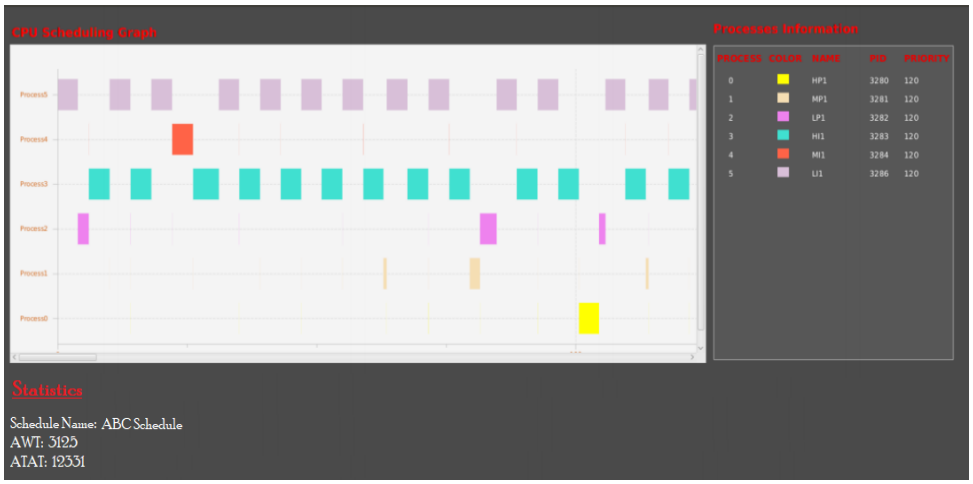
For Each Process you need to receive the following parameters from the user:

- Process Name
- Process Color(Graphical Representation)
- Process Arrival Time
- Process Burst Time
- Process Priority Number

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**)
- **BOUNS:** graphical representation of Processes execution order (Example of Graphical representation)



- The assignment is submitted in group of maximum 5 students.
- **If one student of the team didn't answer well in the discussion slot then his/her teammates will get the mark of this student.**
- Late submission is not allowed

Grading Criteria
BOUNS (10 grades)

	Non preemptive Shortest- Job First (SJF) Scheduling	SRTF Scheduling	Priority Scheduling	AG Scheduling	Grade
Processes execution order	2.5	3.5	3.5	8	**17.5**
Waiting Time for each process	2.5	3.5	3.5	8	**17.5**
Turnaround Time for each process	1	1	1	3	**6**
Average Waiting Time	1	1	1	3	**6**
Average Turnaround Time	1	1	1	3	**6**
Print all history update of quantum time for each process (**AG Scheduling**)	0	0	0	7	**7**
graphical representation	2.5	2.5	2.5	2.5	**10**
Grade	**10.5**	**12.5**	**12.5**	**34.5**	**70**