OS Scheduler

The OS scheduler is a crucial part of the operating system responsible for CPU scheduling. This involves selecting a process from the ready queue and assigning the CPU to it based on specific criteria, primarily the chosen scheduling algorithm and whether it is preemptive or non-preemptive. The dispatcher, a key component of the OS scheduler, handles the actual allocation of CPU resources to the selected process.

For this project, an OS scheduler must be implemented using various scheduling algorithms covered in the course. The project should include two main components: a process generation module, which accounts for 10% of the grade, and a scheduling module, which constitutes 80% of the grade.

Process Generator Module

Each process has a set of parameters such as Arrival Time, Burst Time, and Priority. These parameters must be randomly generated following the specified distribution below to ensure your OS scheduler works correctly under any test case.

Arrival Time	Normal Distribution
Burst Time	Normal Distribution
Priority	Poisson Distribution

To implement this, you must enter an input text file containing number of processes to generate data needed to create the distributions (mean and standard deviation, and lambda in case of poison). The format in the text file is:

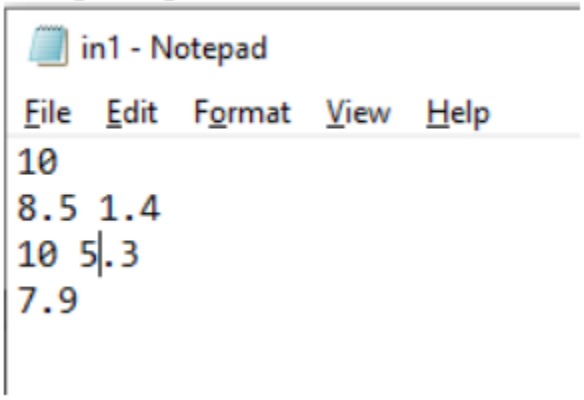
Line 1: number of processes

Line 2: mean and standard deviation for arrival time

Line 3: mean and standard deviation for Burst Time

Line 4: Lamda for Priority

Sample input text file:



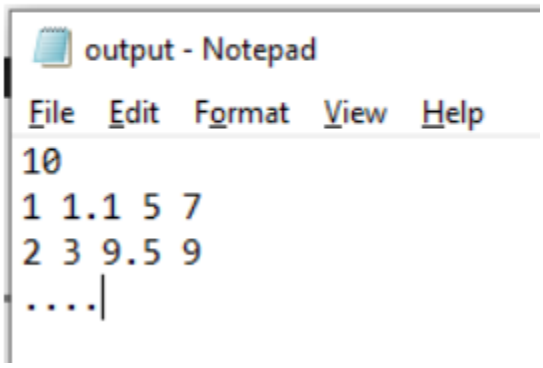
After reading this input text file, it should be slices using appropriate string manipulation provided by your chosen programming language (use white spaces as delimiter and read each line on its own). This should output another text file following the format:

Line 1: Number of processes

Line 2: process ID arrival_time burst_time priority

All following lines follow suit.

Output sample:



OS Scheduler

The second module is the core of the project. The OS scheduler is responsible for creating a schedule by taking the output file of the process generator and scheduling whichever of them in the Ready Queue. You can use whatever architecture you like, whether all processes are checked for execution at the same time based on a counter acting as a "clock tick" or if you want to implement a function that populates a Ready Queue that could take

whatever data structure you deem fittest each clock tick. Your scheduler should implement the following algorithms:

1- Non-Preemptive Highest Priority First (17.5%)

2- First Come First Serve (17.5%)

3- Round Robin (22.5%)

4- Preemptive Shortest Remaining Time First (22.5%)

The final portion of the grade is based on how effectively you demonstrate the impact of each scheduling algorithm on the same input. You must clearly present the turnaround time, waiting time, and average values for each algorithm, along with an analysis of which algorithm performs best for each input case.

Additionally, a **bonus** will be awarded for visualizing the results using graphs. The more polished and well-structured your output is—and the more advanced skills you incorporate, such as developing the project as a website or application—the higher your chances of earning extra credit.

Assumptions:

- Any tie between processes should be broken by order of processes.e.g. if P1 and P2 have the same arrival time and same priority, P1 should execute.
- Assume the greatest priority number is the highest
- Assume no processes will wait for an event or request an I/O. i.e. no processes will enter the waiting state.

Deliverables

1- Samples of Input and Output files.
2- Source codes.
3- Executable file
4- Report your findings.

Report Guidelines

- Your report should have a cover page listing the course name, instructor, TA, team's names, and IDs and submission date.
- Your report should have a list of figures and a table of contents.
- The first page after the tables should have a table of listing each team member and their tasks (workload distribution).
- Any further assumptions made should be included in the report

- Any dead-end reached, in case of failing to implement an algorithm should be discussed in your report.
- You should include screenshots of any schematic, even free hand designs you have used to visualize and plan your projects, but this is not obligatory.
- Finally, comment on your findings and conclude your work.

Project Guidelines

- The maximum number of members in each team can't be more than 4 members.
- All team members must have the same TA.
- Project deadline: 10-4-2025 and discussion will be held depending on your TA instructions.
- GUI is a must.
- No submissions will be accepted after the deadline
- No tolerance for Plagiarism.
- Use any programming language you can implement with.