I. Problem Statement

This project will involve implementing a **preemptive** priority-scheduling algorithm, which schedules processes in order of priority and uses FCFS scheduling for processes with the same priority. Priorities range from 1 to 5, where a lower numeric value indicates a higher relative priority. Assume the overhead of the Context Switch (CS) is 1 ms. Your program should show the following menu:

1. Enter process' information.
2. Report detailed information about each process and different scheduling criteria.
3. Exit the program.

Note*: If a new process arrives at the same time that a process releases the CPU, then the new process will be added to the ready queue first.*

II. Project Description

In this project, the students will write a program to schedule processes in order of their priority according to the **preemptive** priority-scheduling algorithm, and FCFS will be used for processes with the same priority and also the program provides a report about processes and different scheduling criteria. The report should provide detailed information about each process, which includes [process ID, priority, arrival time, CPU burst time, starting and termination time, turn around time, waiting time, and response time] and also report [average turnaround time, average waiting time, and the average response time] of all the processes in the system. The program should behave as follows:

1. The program will prompt the user to enter the number of processes (P).
2. The program will prompt the user to enter the priority, arrival time, and CPU burst of each process.
3. The program will create a PCB array of size P and initialize each process PCB (object) attributes:
    - Process ID: In the form of "PN", where N represents the process number. (Assume the numbering starts from 1)
    - Priority of a process: priorities range from 1 to 5, where a lower numeric value indicates a higher relative priority.
    - Arrival time: The time at which the process enters into the ready queue.
    - CPU burst: The amount of time a process needs to execute.
    - Start and termination time: The start time of process execution and the completion time.
    - Turn around time: The amount of time to execute a process.
    - Waiting time: The amount of time a process has been waiting in the ready queue.
    - Response time: The amount of time it takes from when a request was submitted until the first response is produced.
4. The program will schedule processes execution in the CPU and output on the console a report of each process in the PCB array with the following information, and then will write them to a text file:
    - Process ID.
    - Priority.
    - Arrival time.
    - CPU burst.

- Show the scheduling order of the processes including the CS using a chart. Example: [P1 | CS | P2 | CS | P1].
- Start and termination time.
- Turn around time.
- Waiting time.
- Response time.
- Average [Turn around time, Waiting time, and Response time] for all processes in the system.

The program will prompt the user to input her choice [1, 2, or 3] based on the list mentioned in the *problem statement* section above.

1. If the user selects option 1, the user should be requested to enter the total number of processes in the system and then enters each processes' information, namely, the process priority, the arrival time and the CPU burst time.
2. If the user selects option 2, your program will display detailed information about each process in the system as described above and different scheduling criteria on the console, and then write these information to the output file (*Report.txt*).
3. If the user selects option 3, you should exit the program.

III. Deliverables

The Team Leader is required to submit a single zip folder on the behalf of the group containing the following.

1. Program code in softcopy**. Java** language should be used.
2. A Read Me file in PDF containing the following:
   a. Student names and IDs highlighting Team Leader.
   b. Table showing task distribution. *There should be a clear and practical distribution of tasks.*
   c. Screen shots showing sample input/output so that:
      i. The FCFS condition is demonstrated (i.e., some processes have same priority)
      ii. Preemption is demonstrated (a newly arriving process has a higher priority than the currently executing process)

      In addition, a screen shot of the report (*Report.txt*) should be included.
   d. Student reflection on the simulation. This is a maximum of one paragraph evaluating the performance of the scheduling algorithm and reflecting on the results of the simulation. Students may provide suggestions for improving the performance.
   e. Student peer evaluation form (see Team Work Evaluation table below).