Project 3: CPU Scheduling Algorithms

1. Assignment

In this project you are asked to implement the following 4 CPU scheduling algorithms:

1. **Non-Preemptive** First Come First Served (fcfs)
2. **Preemptive** Shortest Remaining Time First (srtf)
3. **Preemptive** Priority Scheduling (pri)
4. Round Robin Scheduling (rr)

2. Input File

The input to your program will be a test file, where each line contains information about one process in a comma separated values (CSV) format as follows:

Process ID,Arrival Time,Burst Time,Priority

Here is an example test file (test0.txt) with the corresponding process information displayed in a table format:

test0.txt
```
1,0,8,4
2,2,5,2
3,3,4,1
4,8,2,3
5,10,1,1
```

Process	Arrival Time	Burst Time	Priority
1	0	8	4
2	2	5	2
3	3	4	1
4	8	2	3
5	10	1	1

We guarantee that the arrival time of the given processes will be in **non-decreasing** order. That is, for two processes “p” and “q”, if p < q, then arrival time of “p” <= arrival time of “q”. Furthermore, “**priority**” column will be used only for the **preemptive Priority Scheduling** algorithm.

3. Project

Your program will first read in the input file and then simulate the behavior of scheduler supplied from the command file. During the simulation, you will have to record the finish time of each process and at the end, your program will print out the following information for each process in the order of process id in CSV format to the screen as follows. The table on the right shows the meaning of each column in your CSV output.

Output
```
1,0,8,8
2,2,13,11
3,3,17,14
4,8,19,11
5,10,20,10
```

Process	Arrival Time	Finish Time	Turnaround Time
1	0	8	8
2	2	13	11
3	3	17	14
4	8	19	11
5	10	20	10

The output given above belongs to the result you will obtain with the **non-preemptive FCFS** scheduling algorithm, where the next process at the **head of the ready queue** is scheduled when the running process terminates.

Shortest Remaining Time First is a **preemptive** scheduling algorithm in which the scheduler preempts the running process if the newly arriving process has a **shorter burst time**. When selecting the next process to schedule after the running process terminates, the scheduler uses the following logic: T**he process with the shortest burst gets selected first**. If there is more than one process with the same shortest burst, then the process with the **smaller id** is selected.

Priority Scheduling is a **preemptive** scheduling algorithm in which the scheduler preempts the running process if the newly arriving process has a **higher priority**. *Smaller values indicate higher priority*. When selecting the next process to schedule after the running process terminates, the scheduler uses the following logic: T**he process with the highest priority gets selected first**. If there is more than one process with the same priority, then the process with the **smaller id** is selected.

Round Robin Scheduling is a scheduling algorithm where the scheduler schedules the processes in FCFS order but runs each process for at most a user-supplied "**quantum**" amount of time using time slicing. If the running process does not release the CPU when its time quantum expires, the scheduler preempts the process and moves it to the end of the ready queue. If the arrival of a process and the expiration of the time quantum of the running process **happen at the same time**, then you must first append the newly arriving process to the end of the ready queue and then preempt the currently executing process and move it to the end of the ready queue.

4. Testing

Name your program **schedule**. Your program will be run with the following command line arguments:

```
% schedule <test file name> <scheduling algorithm> <time quantum if rr is being used>
% schedule test0.txt fcfs
% schedule test0.txt srts
% schedule test0.txt pri
% schedule test0.txt rr 4
% schedule test0.txt rr 2
```

The first argument to your program is the name of the test file, and the second argument is the scheduling algorithm you must use, which will be one of the following: "fcfs", "srtf", "pri" or "rr". If the scheduling algorithm is "rr", then the user must provide an additional command line argument specifying the time quantum. You must implement the project as described above. Failure to observe the rules will result in loss of points.

When you print the output to the screen, make sure that the output for each process is written to a separate line in CSV format as described in section 3. To test for correctness, I will simply compare the output from my program to the output from your program using the Linux "**cmp**" utility. For example, if my program output is "out1.txt" and your program output is "out2.txt", I will run the following Linux command:

```
% cmp out1.txt out2.txt
```

If the files are the same, **cmp** terminates without printing any output. If there are discrepancies between the files, **cmp** prints the lines where they are different. For you to test your code, I have provided you with two test files: test0.txt and test1.txt and also the expected output for each test file for each of the scheduling algorithms. Make sure that your program's output matches mine. Otherwise, I will assume that your program does NOT work correctly.

5. Submission

Submit your application source code via Brightspace. This project is due on Sunday 5/11/2025 11:59pm. You can implement your project in either **C++**, **Java** or **Python**. But in all cases, it must conform to the specifications given in this document.