CPU Scheduler Simulator (20251225)

Objectives:

In this project, it is expected to develop a CPU scheduler application that simulates many different scheduling algorithms

Inputs must be taken from text file with command arguments. And results also should be written to output file and screen.

Recommendations & Rules:

- The application must be written in C or C++ programming language and must be compiled on POSIX complaint systems. We prefer GNU/Linux Operating Systems.
- Arrays usage is prohibited.
- Don't use any special library for protocols when coding. Write your own functions if it's necessary. Only exemption is data structure functions that given in Data structures lecture.
- Github template codes must be used.
- You should commit changes on github system. Usually at least one time a day you should commit your code on github.
- Unless otherwise explicitly specified, all written assignments or code that is submitted is to be entirely the student's own work. Using any code or copying any assignment from others is strictly prohibited without advance prior permission from the instructor.
- All students' work is their own. Students who do share their work with others are as responsible for academic dishonesty as the student receiving the material.
- Students who encourage others to cheat or plagiarize, or students who are aware of plagiarism or cheating and do not report it are also participating in academically dishonest behavior.
- In this case (academically dishonest behavior), students to be punished with no grade.

Project Details:

1. Program must get two command arguments. First argument specifies input text file and second specifies output text file. Command should be like following format. (File name can be anything so please consider them while you implemented code)

```
student@db:~$ ./cpe351   input_file.txt  output_file.txt
```

2. Input file should have **four** columns. Number of rows may not **limited**. Input file must be text and fields should delimit by ":" character. Below table represent structure of input file.

Burst Time	Priority	Arrival Time	QueueID
10	1	0	0
5	0	1	0
3	2	2	0
2	1	3	0
3	1	0	1
10	0	1	1
2	2	2	1

Some data fields may not use at decision process so you can omit the values. But every process should have four of these values. At the Appendix section input.txt file given as example.

3. Developed program should support following CPU scheduling algorithms:
   a) First Come First Serve
   b) Shortest Job First (Non-preemptive)
   c) Priority (Non-preemptive)

4. After application processed input file it must be dump processed value for each CPU scheduling algorithms to specified output file delimited with ":". Output should be following format:

QueueID	Scheduler Algorithm	WT1	WT2	WT3	WT4	AWT
0	1	0	9	13	15	9.25
0	2					
0	3					
1	1					
1	2					
1	3					

CPU Algorithm values description as follow:

- 1: First Come First Serve
- 2: Shortest Job First (Non-Preemptive)
- 3: Priority (Non-Preemptive)

Number of rows which belongs same QueueID in input file define number of processes that processed by your application. So, columns (WT1, WT2… WTn) may change according to number of rows in input file. Mind that it is not require to have same number of processes belongs to same QueueID. And last column at output file must be Average Waiting Time. AWT fractional part should contain maximum **two** digits.

If you can't implement any of Scheduler algorithms, output row must have Waiting Times and AWT value as "0". Example output file given in Appendix section.

APPENDIX

File input.txt sample content

```
10:1:0:0
5:0:1:0
3:2:2:0
2:1:3:0
3:1:0:1
10:0:1:1
2:2:2:1
```

output.txt/Screen output

```
0:1:0:9:13:15:9.25
0:2:0:14:10:7:7.75
0:3:0:9:15:12:9
1:1:0:2:11:4.33
1:2:0:4:1:1.66
1:3:0:2:11:4.33
```