First assessed lab CITS2002 - 10%
Due Date : 5/9/2025, 11:59 pm (via LMS)

Note:

- This lab has to be completed individually. The lab is very simple, and no external help including from LLM is allowed. We will check for this.
- This lab has to be done using only arrays allocated in the code. Please do not use any dynamic memory allocation even if you know about that. There will be a future assessed lab where we will use dynamic memory allocation.

Description:

The aim of this lab is to simulate process scheduling by the operating system. You will be given a list of processes and the execution time for each process. The time quantum in the system is 10 milli seconds (ms). That means a process with execution time 100 ms needs to be scheduled for 10 time quanta for completion.
However, each process makes some I/O request when it is executing. It makes an I/O request exactly after 5 ms when it is scheduled to run. The process is blocked and the next process is given the CPU, and when a process is given the CPU, it is always given the full time quantum of 10 ms. The next process is given a time quantum if the current process terminates within its time quantum. A process can make at most 10 I/O requests, but some processes may make less than 10 I/O requests. The details of the processes will be given in a file calles `input`. A line of the file will look like this:

```
A 90 3 5 6
```

The name of the process is `A`, it needs 90 ms to complete, and it makes three I/O requests during its time quanta $3, 5$ and 6. There are in all 10 processes and so the file will have 10 such lines.

Your tasks:

- As this is a simulation, there is no actual clock. You have to keep the time a process has completed in a variable, let us say `cpuTime`. If a process is scheduled to run, you can just add 10 to cpuTime and it completes the time quantum. If it makes an I/O request, you add 5 to cpuTime and the process is blocked. A process eventually completes execution when its cpuTime becomes its total execution time. For example, for process `A` in the above example, it completes when its execution time reaches 90. When a process completes its execution within a time quantum, the next process (if any) is given the next time quantum.
- You don't have to implement any `ready to run` or `blocked` queues. Simply go through the processes from 0 to 9 (there are 10 processes) and back to 0 again. A process completes its I/O request by the time it is scheduled again.

- You have to store the total time elapsed from the start of the simulation in a global variable, or in a variable in the `main` function.
- Eventually you have to print the information when each process terminates according to this global time. For example:

```
A 115
B 125
C 135
```

- You have to use an array of structures for storing information of the 10 processes. The members of the structure will be name of a process, its total execution time, how long it has executed until now, an array of its I/O requests etc.
- The `fgets` function can be used for reading from files. `sscanf` can be used for extracting the individual components of a line. `sscanf` has a return value that indicates how many components from a line it has successfully read. You can always read 12 components (name, completion time and 10 possible I/O requests) and then store the correct values checking the return value of `sscanf`.
- I will keep some examples of input and output within a few days.

Marking scheme:

- Correctness of program - 5 marks
- Correct reading of file and processing inputs - 3 marks
- Correct data structures - 2 marks