Second assessed lab CITS2002 - 10%
Due Date : 19/9/2025, 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 is a modified version of the first assessed lab. Please read the description of the first assessed lab carefully, the modifications are outlined below.

Description:

The aim of this lab is to simulate process scheduling by the operating system as in the first lab, but the setting is more general.

- The number of processes was fixed (10) in the first lab. But there may be a variable number of processes in the input file, for example, 5, 10, 15 etc. Your program will be tested with different number of processes in the input file.
- All memory allocation was done in the first assessed lab using fixed size arrays. All memory allocation in this lab must be dynamic using `malloc()`.
- Process names were single letters in the first assessed lab, process names can be strings of varying lengths in this lab.
- A process could get an I/O interrupt and get blocked exactly in the middle of a time quantum in the first assessed lab, a process can get an I/O interrupt anytime during a time quantum in this lab.
- I got many emails asking this question on the first assessed lab, so here is a clarification. If a process gets an I/O interrupt and gets blocked, the next time quantum of the CPU starts immediately (at the next time step), the CPU does not sit idly for the remaining part of the time quantum.
- A time quantum remains the same as in the first lab, 10 time steps.
- A sample file:

```
A1 90 3 7 4 2 5 1
Process2 100 3 1 4 3 6 9 7 4 9 2
```

- **Explanation:** The first string is the name of the process, the number next to it is the total number of time steps a process must run to complete execution.

  The numbers after that come in pairs, for example, 37 for `A1` means `A1` is interrupted after 7 time steps when it is scheduled for the third time. Similarly, `A1` is interrupted after 2 time steps when it is scheduled for the fourth time, etc.

Your tasks:

- You must not allocate any fixed size array in your code, all arrays must be allocated dynamically.
- You don't have to implement any `ready to run` or `blocked` queues. Simply go through the processes from 0 to n (there are n processes) and back to 0 again. A process completes its I/O request by the time it is scheduled again.
- Eventually you have to print the information when each process terminates according to this global time. For example (an arbitrary example):

```
A1  115
Process2  125
```

- 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 - 2 marks
- Correct data structures - 3 marks