De La Salle University
College of Computer Studies
Software Technology Department

INTROOS: Introduction to Operating Systems
Lab Exer: Queues and Scheduling Algorithms

Expected Learning Outcomes:

LO1. Outline the different components of an operating system.
LO2. Implement scheduling algorithms across multiple external job text files.

Instructions:

- Preferably, the programs should be coded in C. However the use of C++, C# and Java are encouraged provided that upon submission, the executables (in .exe or .jar) should be submitted as well.
- The program should be labelled following the format: SURNAME-FirstName_Exer1 (example: DEJA-Jordan_Exer1.c) .
- Submit your works, screenshots and log files in via email following the given details:
  - **Subject**: [INTROOS] Lab Exer 1 : Surname
  - **Body**: Full name.
  - **Attachments**: program source code, screenshot of each algorithm displaying gantt charts, log files.
  - **OPTIONAL**: if you have you have a git hub account, commit your submissions to this link: https://github.com/jrdndj/INTROOS-S11-T21516/tree/submissions
- Use proper identifier names and coding standards.

Tasks:

1. Your program should be able to simulate the ready queue of process by declaring an array or a linked list of records, with the following structure:

   PCB

PID
State
CPU_burst
Arrival_time

   PID refers to Process ID, should be in int
   State of the process, either ready, running, waiting, terminated.
   CPU_burst refers to the amount of time to complete process, usually unsigned int
   Arrival_time refers to time of arrival, in unsigned int

2. A text file (jobs.txt) contains the set of jobs to be executed

   ```
   A 0 7
   B 2 4
   C 4 1
   D 5 4
   ```

   *process ID is labelled from A to Z
   *second digit refers to the burst time of the process
   * third digit refers to the arrival time of the process

3. Simulate the First-Come-First-Served (FCFS) scheduling algorithm using the queue data structure you have designed. Given the same input, the output should have the following sequence:

   ```
   AAAAAAABBBBCDDDD
   ```

4. (Extra Credit) Compute for the individual waiting time for each process as well as the average waiting time for the FCFS scheduling algorithm. The output should display something similar to as seen below:

   ```
   Gantt Chart ( ex: AAAAAAABBBBCDDDD )
   Average Waiting Time (ex. WT( A ) = 0 ;  WT( B ) = 5  ;  WT( C ) =
   7  ;  WT( C ) = 7  ;  AWT = 4.75
   ```

(Even extra credit). These details should be written in a text file called FCFS.log.

5. Another text file (RRjobs.txt) contains the set of jobs to be executed

```
A 00 45
B 03 23
C 12 08
D 21 12
```

*process ID is labelled from A to Z
*second number refers to the burst time of the process
* third number refers to the arrival time of the process

6. Simulate the Round Robin (RR) scheduling algorithm given where the value for time quantum is *09*. Display the Gantt chart, average waiting time and store these details in an external log file (RR.log) .

7. This time, simulate the Shortest-Job-First (SJF)/Shortest-Time-Remaining-First (SRTF) scheduling algorithm given the inputs from a text file (SJFjobs.txt). Display the gantt chart, the average waiting time and have these written in an external log file (SRTF.log).

```
A 0 7
B 2 4
C 4 1
D 5 4
```