# COEN 346
# Programming Assignment #2

## Simulating Round-Robin Process Scheduling

- ✓ **Problem statement:** Implement the simulation of a process scheduler that is responsible for scheduling a given list of processes. The scheduler is running on a machine with one CPU. The scheduling policy is a type of non-preemptive round-robin scheduling and works as follows:
  - Scheduler works in a cyclic manner, i.e. it gives the CPU to a process for a quantum of time and then get the CPU back.
  - The quantum for each process is equal to 10 percent of the remaining execution time of the process.
  - Each process comes with its own arrival time and burst time.
  - Each time, the scheduler gives the CPU to a process (say P1) that has the shortest remaining processing time, but this should not starve other processes in the queue and which are ready to start. These processes should be allocated to the CPU before it is given back to P1, i.e. include some fairness for long jobs already in the queue.
  - In the case that two or more processes have equal remaining time for completion, the scheduler gives priority to the older process (i.e. process that has been in the system for longer time).

  The simulation should determine and print out the time each process has spent in the waiting queue.

- ✓ The input consists of a file named **"input.txt"** and the output should be written to **"output.txt"**.

- ✓ **Input:** Each line of the input file contains information related to one process. The first column is the arrival time (ready time) of the process and the second column shows the required execution time for the process.

```
Sample "input.txt":
              1      5
              2      3
              3      1
```

- ✓ **Output:** Set of strings indicating events in the program including:
  - **Start and End of each process:** E.g. ***Time: 2, Process 1: Started/Finished***. "Time: 2" indicates the occurrence time of the event in seconds. Moreover "Started" or "Finished" indicates the type of event.
  - **Start and End of each time slice:** This event happens when the scheduler pauses or resumes the execution of a process (i.e. load or remove the process from CPU). E.g. ***Time: 5, Process 2:***

***Paused/Finished***. "Time: 5" indicates the occurrence time of the event in seconds, "Process 1" indicates the process related to this event. "Paused" or "Finished" indicates the type of event.

- **Waiting time for each process:** For example, Process 2 waits 2 seconds in the waiting mode, in total, that is (Time2 – Time1) + (Time4 – Time3) = (2-1) +(4-3) = 1+1= 2.

Sample "output.txt":

```
Time 1, Process 1, Started
Time 1, Process 1, Resumed
Time 2, Process 1, Paused
Time 2, Process 2, Started
Time 2, Process 2, Resumed
Time 3, Process 2, Paused
Time 3, Process 3, Started
Time 3, Process 3, Resumed
Time 4, Process 3, Paused
Time 4, Process 3, Finished
Time 4, Process 2, Resumed
Time 5, Process 2, Paused
Time 5, Process 2, Resumed
Time 6, Process 2, Paused
Time 6, Process 2, Finished
Time 6, Process 1, Resumed
Time 7, Process 1, Paused
Time 7, Process 1, Resumed
Time 8, Process 1, Paused
Time 8, Process 1, Resumed
Time 9, Process 1, Paused
Time 9, Process 1, Resumed
Time 10, Process 1, Paused
Time 10, Process 1, Finished
----------------------------
Waiting Times:
Process 1: 4
Process 2: 2
Process 3: 2
```

Note that in the example above, the quantum is set to 1 second.

✓ **Implementation Requirements:**

- The processes (both scheduler and processes) **must** be simulated using threads. The scheduler should be able to resume each process and only one process should be running at a time (because the simulated system has only one CPU). This means that the simulated process (P1) should suspend its execution and give the CPU back to the scheduler thread, then the scheduler resumes the next simulated process (e.g. P2) based on the discussed scheduling policy. Each process is responsible to keep track of its total execution and inform the scheduler thread when it is finished. This process continues until all processes are done.
- The program must work with arbitrary number of threads.
- Note that the implementation should be done in Java or C++.
- The assignment should be done in a **group of two students**.

- The deliverable must be a .zip file which consists of:
  - A well-commented code,
  - The executable file of the assignment (e.g. example.exe).
  - A report document. Do not include your code in your report
- The report should be at least two pages with the following sections:
  - Name of group members
  - High level description of the code (description of the methods/functions/threads/data structures and the flow of the program).
  - A detailed conclusion, discussing you experience with using thread for simulating shortest job remaining time scheduling (e.g. is the simulation accurate?). You also need to analyze your program and discuss its drawbacks and advantages compared to pure round robin technique among all the processes independently of the processing time.

This assignment will take 40% of your total mark for programming assignments. Also for this assignment 80% of the mark is dedicated to your code and 20% to your report.

The code and the report should be submitted through EAS website (https://fis.encs.concordia.ca/eas/). The deliverable should be submitted through EAS before Monday March 09th at 11 pm.