# School of Electrical Engineering and Computing
# COMP2240/COMP6240 - Operating Systems

# Assignment 1 (10%)

Submit using Blackboard by **11:59 pm, Friday 6th September 2019**

## Task:

Write a program that simulates **First Come First Serve** (FCFS), **Round Robin** (RR), **Feedback constant** (FB) and **Narrow Round Robin** (NRR) scheduling algorithms. For each algorithm, the program should **list the order and time** of the jobs being processed and **compute waiting time** and **turnaround time** for every job as well as the **average waiting time** and **average turnaround time**.

The average values should be consolidated in a table for easy comparison (*sample outputs are available through Blackboard*).

Two sample input data sets and the corresponding outputs have been supplied. Additional datasets will be used to test your program. The format of the input data will be the same as in the supplied sample files.

Each input data set contains the following information (*check the sample input files for exact format*):

1. Time for running the dispatcher (`DISP`)
2. For each process: process id (`ID`) , arrival time (`Arrive`), service time (`ExecSize`)
    a. It can be assumed that process `P_i` will always arrive before or at the same time of process `P_(i+1)`

**Dispatcher:** It is assumed that the dispatcher runs to select the next process to run. The dispatcher should behave as follows:

(i) The time to run the dispatcher is fixed and taken as input (`DISP`) from the input file. No other time is wasted in switching between processes other than this.

(ii) If there is only one process running in the processor and no other process is waiting in the ready queue then there is no need to switch the process and the dispatcher will NOT run. For example, in RR scheduling if process P1 is running in the CPU and no other process is waiting in the ready queue then P1 will continue even after its time quantum expires – no need to interrupt P1 to send it to ready queue after its time quantum expires and then run the dispatcher to reload P1 from the ready queue.

(iii) If the dispatcher starts at $t_1$ and finishes at $t_2$ (*i.e. time to run the dispatcher* is $t_2$-$t_1$) then in that run it will choose from the processes that have arrived at or before $t_1$. It will not consider any process that arrives after $t_1$ for dispatching in that run.

(iv) If a process P1 is interrupted at $t_1$ and another process P2 arrives at the same time $t_1$ then the newly arrived process P2 is added in the ready queue first and the interrupted process P1 is added after that.

(v) If two processes Px and Py have all other properties same (*e.g. arrival time, priority etc.*) then the tie between them is broken using their ID i.e. Px will be chosen before Py if x<y.

Some details about the scheduling algorithms are as follows:

**FCFS:** Standard FCFS scheduling algorithm.
**RR:** Standard RR scheduling algorithm with time quantum of **4 milliseconds**.
**FB:** Standard FB constant scheduling algorithm with time quantum of **4 milliseconds** and a **6 level priority** (0 is highest priority and 5 is lowest priority)
**NRR:** NRR Scheduling is a variant of the standard Round Robin (RR) scheduling in which each process has its own time quantum q. Each process starts with **q = 4ms** and **q is decreases by 1ms each time the process goes through the round robin queue, until it reaches a minimum of 2 ms**. Thus, long jobs get decreasingly shorter time slices.

## COMP6240 Students:

In addition to the above simulations, you are to implement a RR scheduling simulation for a **dual processor system**. Assume that two processors share the same ready queue and the **time quantum** for processor 1 is **2 milliseconds** and for processor 2 is **3 milliseconds**, and that the system starts loading processes from processor 1. For dispatching a process, the dispatcher will run on the processor that is idle. If both processors are idle at the same time and there are jobs in the ready queue then at first the dispatcher will run on processor 1 and then it will run on processor 2 to dispatch processes for them respectively. Output the same data as required for the other simulations.

## Programming Language:

The programming language is Java, versioned as per the University Lab Environment (**currently a subversion of Java 1.8**). You may only use standard Java libraries as part of your submission.

## User Interface:

There are no marks allocated for using or not using a GUI – the choice is yours.

## Input and Output:

Your program will accept data from an input file of name specified as a command line argument. The sample files `datafile1.txt` and `datafile2.txt` (containing the set1 and set2 data) are provided to demonstrate the required input file format.

Your submission will be tested with the above data and will also be tested with other input files.

Your program should output to standard output (*this means output to the Console*). Output should be strictly in the order FCFS, RR, FB, NRR, Summary.

The sample files `datafile1_output.txt` and `datafile2_output.txt` (containing output for `datafile1.txt` and `datafile2.txt` respectively) are provided to demonstrate the required output (and input) format which **<u>must be strictly maintained</u>. <u>If output is not generated in the required format then your program will be considered incorrect.</u>**

Two Gantt's charts are provided to explain the corresponding behaviour of different algorithms and the dispatcher for the two sample data files.

## Deliverable:

1. Your submission will contain your program source code, documentation (below) and a `readme.txt` (containing any special instructions required to compile and run the source code) in the root of the submission. These files will be zipped and submitted in an archive named **`c9876543.zip`** (where **`c9876543`** is your student number) – do not submit a `.rar`, or a `.7z`, or etc.

2. Your main class should be **`A1.java`** you program will compile with the command line **`javac A1.java`** and your program will be executed by running **`java A1 input.txt`**.

3. Brief **1 page** (*A4*) report, reviewing the results from your program and any interesting observations. Specifically, write a note about the relative performance of the algorithms based on your implemented versions of the algorithms.

4. Completed Assignment Coversheet.

**NOTE**: Assignments submitted after the deadline (**11:59 pm Friday 6th September 2019)** will have the maximum marks available reduced by 10% per 24 hours.

## Mark Distribution:

Mark distribution can be found in the assignment feedback document (`Assign1Feedback2240.pdf` and `Assign1Feedback6240.pdf`).

SENG2240 Staff<br>
**v1.0** (*2019-08-15*)<br>
**v1.01** (*2019-08-20*) – corrected typo in and added clarity to the **I/O** and the **Deliverables** section.