THE UNIVERSITY OF NEWCASTLE AUSTRALIA

School of Information and Physical Sciences
Semester 2, 2023 - COMP2240/COMP6240 - Operating Systems

Assignment 1 (10%)

Submit using Canvas by **11:59 pm, Friday 25th August 2023**

1. Introduction

Assignment 1 will focus on your understanding and ability to successfully implement, execute and compare the results of a number of scheduling algorithms.

2. Assignment Task

Write a program that simulates **First Come First Serve (FCFS)**, **Shortest Process Next (SPN)**, **Preemptive Priority (PP)** and **Priority Round Robin (PRR)** scheduling algorithms.

For each algorithm the program should list the order and time of the processes being loaded in the CPU, compute turnaround time and waiting time for every process, as well as the average turnaround time and average waiting time. The average values will be consolidated in a table for easy comparison (check the sample outputs).

Two sample input data sets and the corresponding outputs have been supplied. Beyond the two sample datasets provided, 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.

2.1. Input Data

Each input data set contains the following information (*check the sample input files for exact format*); times can be considered to just be whole number *'time units'*:

1. Time for running the dispatcher (`DISP:`) which can be zero or a positive integer
2. For each process:
   i. process id (`PID:`) – as a *string* in the form `p`*n* where n is a positive integer 1, 2, 3, … (in order).
   ii. arrival time (`ArrTime:`) – as a non-negative *integer* representing *unit of time*
   iii. service time (`SrvTime:`) – as a positive *integer* representing *unit of time*
   iv. process priority (`Priority`) – as an integer from {0, 1, 2, 3, 4, 5} where 0 is the highest priority and 5 is the lowest priority.

It can be assumed that process P*n* will always arrive before or at the same time of process P*n+1*

2.2. Dispatcher

It is assumed that the dispatcher is executed 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 PRR scheduling if process `p1` is running in the CPU and no other process is waiting in the ready queue then `p1` will continue after its time quantum expires – no need to interrupt `p1` to send it to ready queue after its time quantum expires then run the dispatcher to reload `p1` from the ready queue.

iii. If the dispatcher starts at *t1* and finishes at *t2* (*i.e. time to run the dispatcher is t2-t1*) then in that run it will choose from the processes that have arrived at or before *t1*. It will not consider any process that arrives after *t1* for dispatching in that run.

iv. If a process `p1` is interrupted at *t1* and another process `p2` arrives at the same time *t1* 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 $x<y$.

2.3. Scheduling Algorithm Details

Some details about the scheduling algorithms are as follows:

FCFS: Standard FCFS scheduling algorithm. Process priority is ignored in scheduling.

SPN: Standard SPN scheduling algorithm. Process priority is ignored in scheduling.

PP: Standard preemptive priority scheduling algorithm.

PRR: This is a variant of the standard Round Robin (RR) algorithm. Processes are divided into two priority classes Higher Priority Class (HPC): processes with priority 0, 1 or 2 and Lower Priority Class (LPC): processes with priority 3, 4 or 5. PRR algorithm is exactly same as the standard RR algorithm except each HPC process receives a time quantum of 4 units and each LPC process receives a time quantum of 2 units.

3. COMP6240 Additional Task

In addition to the above simulations, you are to implement the Round Robin 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 time units* and for *processor 2* is *3 time units*.

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 it will dispatch a process in *processor 2*, if there are more processes, it will be dispatched to *processor 1* in the same run of dispatcher. If one processor becomes idle, then the dispatcher will run in that processor and will dispatch a process for it.

Output the same data as required for the other simulations. Additionally, you will need to specify in which processor a job is assigned at a specific time [i.e. instead of **`T1: p1`** use **`P1-T1: p1`** to clarify that the *process 1* (**`p1`**) is running in *processor 1* (**`P1`**) starting at *time 1* (**`T1`**)].

4. Submission Requirements

Your submission must also conform to the follow requirements.

4.1. Programming Language

The programming language is Java, versioned as per the University Lab Environment (*Note this is currently a **subversion of Java 17***). You may only use standard Java libraries as part of your submission.

4.2. User Interface

The output should be printed to the console, and strictly following the output samples given in the assignment package. While there are no marks allocated specifically for the Output Format, **there will be a deduction when the result values and result format vary** from those provided.

4.3. 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. **Hint:** the **Java Scanner Library** is something you will likely want to use here!

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**, **SPN**, **PP**, **PRR**, **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 **must be strictly maintained**. **If**

output is not generated in the required format then your program will be considered incorrect.

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

4.4. Mark Distribution

Mark distribution can be found in the assignment draft marking breakdown documents: `23s2_Assign1_Feedback_2240.pdf` & `23s2_Assign1_Feedback_6240.pdf`.

4.5. Deliverable

The following must be observed in your deliverable:

1. Your submission will contain your program source code with documentation and the report (*below*) 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`** your program will compile with the command line **`javac A1.java`** and your program will be executed by running **`java A1 input.txt`**.

   …where **`input.txt`** can be *any* filename – **do not hardcode filenames or extensions**!

   Note: If your program can not be compiled and executed in the expected manner (*above*) then please add a `readme.txt` (containing any special instructions required to compile and run the source code) in the root of the submission. Please note that such non-standard submissions will be **marked with heavy penalty**.
3. Brief **1 page** (*A4*) report (name as **Report.pdf**), reviewing the results from your program and any interesting observations. Specifically, write a note about the type of testing was done, the relative performance of the algorithms based on your implemented versions of the algorithms, any unexpected behaviour you noticed in any of the algorithms.

4.6. Submission Notes

1. Assignments submitted after the deadline (**11:59 pm 25th August 2023)** will have the maximum marks available reduced by 10% per 24 hours.
2. If your assignment is not prepared and submitted following above instructions then you will lose most of your marks despite your algorithms being correct.
3. If you have been granted an extension, please include a copy of the extension approval in as a **PDF document**, called `Extension.pdf`, and place this in the root of your submission, along with your other documents and code.