School of Electrical Engineering and Computing
COMP2240/COMP6240 - Operating Systems
Assignment 3 (15%)

Submit using Blackboard by **11:59 pm**, **Friday 1st November 2019**

Task:

In Assignment 1, we assumed that the system had an infinite amount of memory. In this assignment, the operating system has a **limited amount of memory**, and this needs to be managed to meet **process demands**. **You will write a program to simulate a system that uses paging with virtual memory.** The characteristics of the system are described below:

- **Memory:** The system has *F* frames available in main memory. During execution, the processor will determine if the page required for the currently running process is in main memory.
    a. If the page is in main memory, the processor will access the instruction and continue.
    b. If the page is not in main memory, the processor will issue a page fault and block the process until the page has been transferred to main memory.
    c. Initially no page is in the memory. *ie*: the simulation will be strictly demand paging, where pages are only brought into main memory when they are requested.
    d. In fixed allocation scheme frames are equally divided among processes, additional frames remain unused.
- **Paging and virtual memory:** The system uses paging (*with no segmentation*).
    a. Each process can have a maximum *50 pages* where each page contains a single instruction.
    b. For page replacement you will need to apply *least recently used* (LRU) and *clock* policy.
    c. Resident set is managed using *'Fixed Allocation with Local Replacement Scope'* strategy. *ie*: you can assume, frames allocated to a process do not change over the simulation period and page replacements (*if necessary*) will occur within the allocated frames.
    d. All pages are *read-only*, so no page needs to be written back to disk.
- **Page fault handling:** When a page fault occurs, the interrupt routine will handle the fault by placing an *I/O request into a queue*, which will later be processed by the *I/O controller* to bring the page into main memory. This may require replacing a page in main memory using a page replacement policy (LRU or clock). Other processes should be scheduled to run while such an *I/O operation* is occurring.
    a. Issuing a page fault and blocking a process *takes no time*. If a page fault occurs, then another ready process can run immediately at the same time unit.
    b. Swapping in a page takes *6 units of time* (if a page required by a process is not in main memory, the process must be put into its blocked state until the required page is available).
- **Scheduling:** The system is to use a *Round Robin short-term scheduling algorithm* with time a quantum of *Q*.
    a. Executing a single instruction (*ie*: a page) takes 1 unit of time.
    b. Switching the processes does not take any time.
    c. If multiple process becomes ready at the same time, then they enter the ready queue in the order of their ID.
    d. All the processes start execution at time *t=0*.
    e. If a process becomes ready at time unit *t* then execution of that process can occur in the same time unit *t* without any delay.

You are to compare the performance of the **LRU** and **Clock** page replacement algorithms for the **fixed allocation with local replacement** strategy.

Please use the basic versions of the policies introduced in lectures.

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.

If you wish to use any language other than the preferred programming language, you must first notify the course demonstrator (Dan).

User Interface:

The output will 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 format varies from those provided.

Input and Output:

Input to your program will be via **command line** arguments, where the arguments are **system configurations** and the **names of files** that specify the execution trace of different processes. All processes start execution at **time *t=0***, and are entered into the system in the **order of the command line arguments** (*with the third argument being the earliest process*).

For example:

```
java A3 30 3 process1.txt process2.txt process3.txt
```

…where **30** is the number of frames (**F**), **3** is the quantum size (**Q**) for Round Robin algorithm, and the other arguments are **text file names** containing page references for each process.

Since we assume that each page contains a single instruction, an execution trace is simply a page request sequence.

For example:
```
begin
1
3
2
1
4
3
end
```

This specifies a process that first reads *page 1*, then *page 3*, and so on until the *'end'* is reached.

For each replacement strategy, the simulation should produce a summary showing, for each process, the total number of page faults, the time the page faults were generated, and the total turnaround time (*including I/O time*). *See samples*.

Sample inputs/outputs are provided in the Assignment Archive. Working of the first example is presented with details of different levels. Your code will be tested with additional inputs. **Your program should output to standard output.**

Output should be **strictly** in the shown output format. **If output is not generated in the required format then your program will be considered incorrect.**

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 **`A3.java`** you program will compile with the command line **`javac A3.java`** and your program will be executed by running **`java A3 30 3 process1.txt process2.txt process3.txt`**, as explained in detail, in the **Input and Output** section above.

3. Brief 2 page (A4) review of **how and why you designed your program**, a **comparison of the page replacement methods** based on the results from your program, and **a personal reflection** on the assignment (*issues*, *challenges*, *discoveries*, *estimates vs reality*, etc).

NOTE: Assignments submitted after the deadline (**11:59 pm Friday 1st November 2019)** will have the maximum marks available reduced by 10% per 24 hours.

Mark Distribution:

Mark distribution can be found in the accompanying assignment feedback document (`Assign3Feedback.pdf`).

COMP2240 Staff
v1.0 (*2019-09-26*)