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

Submit using Blackboard by **11:59 pm**, **Friday 27th October 2017**

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 that simulates a system that uses paging with virtual memory. The characteristics of the system are described below:

- **Memory:** The system has 30 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. I.e. 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. I.e. 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 3.
    a. Executing a single instruction (i.e. a page) takes 1 unit of time.
    b. Switching the processes do 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.

Input and Output

Input to your program should be via **command line** arguments, where each argument is the name of a file that specifies the execution trace of a process. All processes start execution at time 0, and are entered into the system in the order of the command line arguments (with the first argument being the earliest process). For example:

```
mysimulator process1.txt process2.txt process3.txt
```

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

For example: (`process1.txt`)

```
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).

Sample inputs/outputs are provided. 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 (for non-GUI solutions) or to a text area (for GUI solutions). 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.**

User Interface:

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

Programming Language:

The preferred programming language is Java, C (gcc), C++ (g++).
If you wish to use any language other than the preferred programming language, you must first notify the course demonstrator.

Deliverable:

1. Program source code and a README file containing compiler with version number, any special instructions required to compile and run the source code. If programmed in Java, your main class should be c9999999A3 (where c9999999 is your student number) i.e. your program can be executed by running "java c9999999A3". If programming in other languages, your code should compile to an executable named "c9999999A3".

2. Brief 1 page (A4) review of how you tested your program and a comparison of the page replacement methods based on the results from your program and any interesting observations.

Please submit all files and the 1 page report plus a copy of the official assignment cover sheet in a ZIPPED folder through Blackboard. The folder name should be “c9999999A3” and the zip file should be name as c9999999A3.zip (where c9999999 is your student number).

Mark Distribution:

Mark distribution can be found in the assignment feedback document (Assign3Feedback2240.pdf).