Final Project CMSC – OPERATING SYSTEMS

Design and implement a **Demand Paging** virtual memory simulator!
It must be a text based application (NOT a GUI based one).
You can use the **C/C++** or **Java** programming language.
The following algorithms must be implemented: **FIFO**, **OPT, LRU** and **LFU**.
The application must simulate the execution of each of these algorithms on a hypothetical computer having only **N** physical frames (numbered from **0 to N-1, N<8**), assuming that the single process that is running has a virtual memory of **ten** frames (numbered from **0 to 9**). The number N should be a number provided in the command line as an argument.
The algorithms will be simulated based on a reference string (a sequence of pages that are to be accessed) that will be either read from the keyboard or randomly generated.

THE SIMULATION MUST FOLLOW THE ANIMATED EXAMPLES FROM THE ONLINE MODULE 3 AS CLOSE AS POSSIBLE IN ALL ASPECTS !!!

The program should be menu-based and the menu will keep the user in a **loop** containing the following options:

0 – Exit
Will exit the program
1 – Read reference string
A reference string will be read from the keyboard and stored in a buffer. Each value of the reference string will be verified and validated (or rejected).
Using option 1 again will result in overwriting the old reference string.
2 – Generate reference string
A reference string will be randomly generated; the length of the reference string will be given by the user interactively. The string will be stored in a buffer.
Using option 2 more than once will result in overwriting the old reference string.
3 – Display current reference string
Will display the stored reference string; if there is no reference string stored yet, an error message will be displayed.
4 – Simulate FIFO
Will simulate the step by step execution of the FIFO algorithm using the stored reference string; if there is no reference string stored yet, an error message must be displayed.
The user will press a key after each step of the simulation to continue the simulation.
The total number of faults will be displayed at the end of the simulation.
5 – Simulate OPT
Will simulate the step by step execution of the OPT algorithm using the stored reference string; if there is no reference string stored yet, an error message must be displayed.
The user will press a key after each step of the simulation to continue the simulation.
The total number of faults will be displayed at the end of the simulation.
6 – Simulate LRU
Will simulate the step by step execution of the LRU algorithm using the stored reference string; if there is no reference string stored yet, an error message must be displayed.
The user will press a key after each step of the simulation to continue the simulation.

The total number of faults will be displayed at the end of the simulation.

7 – Simulate LFU

Will simulate the step by step execution of the LFU algorithm using the stored reference string; if there is no reference string stored yet, an error message must be displayed.
The user will press a key after each step of the simulation to continue the simulation.
The total number of faults will be displayed at the end of the simulation.

Selecting a different option will result in an error message but the user will NOT exit the loop!

Deliverables:

1. The source code of the project
2. A report document (report.doc/report.pdf/…) containing an introduction and an overview of the project, then a comprehensive description of the design and the implementation of your project.
3. A test document (test1.doc/ test1.pdf/ …) containing screensots that show the execution of the 4 algorithms using the inputs from HW6. Three screenshots are required for each algorithm: one that shows the beginning of the simulation, one in the middle of the simulation and one showing the end of the simulation.
4. A test document (test2.doc/ test2.pdf/ …) containing screensots that show the execution of the 4 algorithms using the following inputs: N=5, ref. string is:
   0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 5
   Three screenshots are required for each algorithm: one that shows the beginning of the simulation, one in the middle of the simulation and one showing the end of the simulation.

Post the source code of your simulator and the test documents under the Final Project assignment.

Grading Components:

- Comments (Max 5%) – All code is accurately and neatly commented
- Error Checking (Max 5%)- All code provides error checking to prevent and catch runtime errors
- Correct implementation of the main program loop (Max 5%)
- Correct FIFO implementation (Max 11%) – FIFO design and implementation correct and complete
- Correct OPT implementation (Max 11%) – OPT design and implementation correct and complete
- Correct LRU implementation (Max 11%) - LRU design and implementation correct and complete
- Correct LFU implementation (Max 11%) - LRU design and implementation correct and complete
- User friendliness (Max 9%) – Project is easy to use and intuitive

- Introduction and Overview (Max 4%) - Report provides an introduction and overview of the implementation
- Project Description (Max 4%) - Report provides a project description that is detailed, accurate and well-written
- FIFO Testing (Max 6%) – FIFO is fully tested against the specified reference strings
- OPT Testing (Max 6%) - OPT is fully tested against the specified reference strings
- LRU Testing (Max 6%) - LRU is fully tested against the specified reference strings
- LFU Testing (Max 6%) - LRU is fully tested against the specified reference strings