CS 4310 Operating Systems
Project #2 Simulating Page Replacement Manager and Performance Analysis

Due: 11/21
(Total: 100 points)

Student Name: ____

Date: ________________________________

Important:
- Please read this document completely before you start coding.
- Also, please read the submission instructions (provided at the end of this document) carefully before submitting the project.

Project #2 Description:

Simulating Page Replacement Manager of the Operating Systems by programming the following three page replacement algorithms that we covered in the class:

a. First In First Out (FIFO)
b. Least Recently Used (LRU)
c. Optimal Algorithm

You can use either Java, or C++ for the implementation. The objective of this project is to help student understand how above three page replacement algorithms operates by implementing the algorithms, and conducting a performance analysis of them based on the performance measure of ***page faults*** for each page replacement algorithm using multiple inputs. Output the details of each algorithm's execution. You need to show what pages are inside the page frames along with the reference string and mark it when a page fault occurred. You can choose your display format, for examples, you can display the results for each reference string in a table format as shown in the class notes. The project will be divided into three phases to help you to accomplish above tasks in in a systematic and scientific fashion: Design and Testing, Implementation, and Performance Analysis.

The program will read in a reference string a file (ReferenceString.txt) – this file will be generated by you. In this project, assume that

(1) the length of the reference string is always 30, e.g. 361724720354720146353214567012
(2) there are 8 pages, from 0 to 7.

Note that you need to generate 50 testing cases, or 50 reference strings of length 30. You are required to run each reference string using three algorithms with *NumberOfPageFrame* (3, 4, 5 and 6) page frames.
A sample input file of having 5 page frames and a reference string 361724720354720146353214567012 (always with length 30) is given as follows:

[Begin of ReferenceString.txt]
NumberOfPageFrame value:
5
Reference String:
361724720354720146353214567012
[End of ReferenceString.txt]

You can implement the algorithms in your choice of data structures based on the program language of your choice. Note that you always try your best to give the most efficient program for each problem.

Submission Instructions:

- ***turn in the following @blackboard.cpp.edu after the completion of all three parts, part 1, part 2 and part 3***
    - ***(1) three program files (your choice of programming language with proper documentation)***
    - ***(2) this document (complete all the answers)***

Part 1
Design & Testing (30 points)

a. Design the program by providing pseudocode or flowchart for each page replacement algorithm.

<insert answers here>

b. Design the program correctness testing cases. Give at least 4 testing cases (with 3, 4, 5, or 6, page frames) to test your program, and give the expected correct output (# of page faults) of the program for each case in order to test the correctness of each algorithm.

<complete the following table>

Testing case #	Input (Input file with a number of page frames value and a reference string)	Expected # of page faults for FIFO (√ if Correct after testing in Part)	Expected # of page faults for LRU (√ if Correct after testing in Part 3)	Expected # of page faults for Optimal Algorithm (√ if Correct after testing in Part 3)
1 (3 page frames)	*NumberOfPageFrame* value: 3 Reference String: <insert answers here>	<insert answers here>	<insert answers here>	<insert answers here>
2 (4 page frames)	*NumberOfPageFrame* value: 4 Reference String: <insert answers here>	<insert answers here>	<insert answers here>	<insert answers here>
3 (5 page frames)	*NumberOfPageFrame* value: 5 Reference String: <insert answers here>	<insert answers here>	<insert answers here>	<insert answers here>
4 (6 page frames)	*NumberOfPageFrame* value: 6 Reference String: <insert answers here>	<insert answers here>	<insert answers here>	<insert answers here>

c. Design testing strategy for the programs. Discuss about how to generate and structure the randomly generated inputs for experimental study later in Part 3.
Hint 1: To study the performance evaluation of the three page replacement algorithms, it is the easiest to use a random number generator for generating the inputs. However, student should store each data set of 50 trails (reference strings) for each of the four frame sizes (3, 4, 5, 6) and use the same data set for running each of the three page replacement algorithms.
Hint 2: The average performance result (number of page faults) of each input data size can be calculated after an experiment is conducted in 50 trails (reference strings) of each page frame size (3, 4, 5, 6). We can denote the run time results (number of page faults) as the set $X = \{x_1, x_2, x_3 \ldots x_{50}\}$ which contains 50 results (number of page faults) for each page frame size, and each x_i is the number of page faults of each trail (reference string number i).

The Average time = $\frac{\sum_{i=1}^{50} Xi}{50}$

<insert answers here>

Part 2
Implementation (30 points)

a. Code each program based on the design (pseudocode or flow chart) in Part 1(a).

   <generate three programs and stored them in three files, needed to be submitted>

b. Document the program appropriately.

   <generate documentation inside the three program files>

c. Test you program using the designed testing input data given in the table in Part 1(b), Make sure each program generates the correct answer by marking a "√" if it is correct for each testing case for each program column in the table. Repeat the process of debugging if necessary.

   <Complete the three columns of the three algorithms in the table @Part 1(b)>

d. For each program, capture a screen shot of the execution (Compile&Run) using one testing case to show how this program works properly

   <Insert totally three screen shots, one for each program, here>

By now, three working programs are created and ready for experimental study in the next part, Part 3.

Part 3
Performance Analysis (100 points)

a. Run each program with the designed randomly generated input data given in Part 1(c). Generate a table for all the experimental results for performance analysis as follows.

number of page frames	Average of the total completion time (FIFO Program)	Average of the total completion time (LRU Program)	Average of the total completion time (Optimal Algorithm)
3 page frames	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*
4 page frames	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*
5 page frames	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*
6 page frames	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*	*50 results (page faults) for 50 trails (reference strings) and its average*

b. Plot a graph of each algorithm, average page fault vs. page frame size (3, 4, 5, 6) and summarize the performance of each algorithm based on its own graph.

<Insert totally three graphs, one for each program, here>

<write a summary>

Plot all three graphs on the same graph and compare the performance (page faults) of all three algorithms. Rank three page replacement algorithms. Try giving the reasons for the findings.

<Insert three-graphs-in-one graph here>

<write about explaining the results>

c. Conclude your report with the strength and constraints of your work. At least 100 words. (Note: It is reflection of this project. If you have a change to re-do this project again, what you like to keep and what you like to do differently in order get a better quality of results.)

<write a conclusion about strength and constraints of your work here.>