# Programming Assignment #3

CSE 3320.001 and CSE 3320.003

## Due: April 13, 2020 05:30PM CDT

## April 13, 2020 11:30PM UTC

## April 14, 2020 04:00AM IST

## April 14, 2020 04:15AM NPT

## Description

Write a program that implements the FIFO, Optimal, MFU, and LRU page-replacement algorithms. Given a page-reference string, where page numbers range from 0 to 9, apply the page-reference string to each algorithm, and output the number of page faults incurred by each algorithm. Write your code so that the number of page frames in the page table can vary from 1 to 10.

You may work in groups of two for this assignment. If you wish to work in a group the group leader must email me by April 3rd, 2020. Make sure to put Assignment 3 and your section number (001 or 003) in the subject line. Only the group leaders needs to submit the assignment.

## 1.0 Functional Requirements

1.1: Your program shall be run with the following:

```
./a.out <filename>
```

Example:

```
./a.out datafile.txt
```

1.2: Your output shall be the number of page faults for each of the four algorithms on a separate line:

```
Page faults of FIFO:    23
Page faults of LRU:     13
Page faults of MRU:     17
Page faults of Optimal: 10
```

Output shall be in alphabetical order of the algorithms.

1.2: The datafile will consist of lines of integers. Each line shall start with an integer denoting the size of the working set. The rest of the line shall be the pages requested.

Given a data file with the single line below:

```
3 5 2 2 0 4 6 4 4 1 3 4 5 0 4 5 6 2 0 0 1
```

This will be parsed as:

Page Table Size = 3
Page Reference String = 5 2 2 0 4 6 4 4 1 3 4 5 0 4 5 6 2 0 0 1

1.3 There shall be no limit to the number of lines in the input data file.

## 2.0 Nonfunctional Requirements

2.1: You may code your solution in C or C++.

2.2: C files shall end in .c . C++ files shall end in .cpp

2.3: Your source files must be ASCII text files. No binary submissions will be accepted.

2.4: Tabs or spaces shall be used to indent the code. Your code must use one or the other. All indentation must be consistent.

2.5: No line of code shall exceed 100 characters.

2.6: Each source code file shall have the following header filled out:

```
/*
 * Name:
 * ID #:
 * Replacement Assignment
 * Description:
 */
```

2.7: All code must be well commented. This means descriptive comments that tell the intent of the code, not just what the code is executing. The following explains the intent:

```
// We are going to move the working_str as we process
// the string the pointer points towards so we need to
// keep track of its original value so we can deallocate
// the correct amount at the end
char *working_root = working_str;
```

The following just describes to code and provides no real value and will earn less credit:

```
// Store the value of the working string pointer
char *working_root = working_str;
```

When in doubt over comment your code.

2.8: Keep your curly brace placement consistent. If you place curly braces on a new line , always place curly braces on a new end. Don't mix end line brace placement with new line brace placement.

2.9: Each function should have a header that describes its name, any parameters expected, any return values, as well as a description of what the function does. For example :

```
/*
 * Function: processSystemTime
 * Parameter(s): time_delta - A float representing the time drift
 * since the system was last queried. Normalized (0-24 hrs) to
 * 0.0-1.0
 * Returns: An integer value of 0 for success and -1 for error
 * Description: Processes the time value drift calculated from
 * the last received satellite update.
 */
int processSystemTime( float time_delta )
```

## Grading

The assignment will be graded out of 100 points. Compiler warnings are there to tell you something is not correct. Pay attention to them and you will save yourself a lot of late nights debugging code. Code that does not compile will earn 0.

## Submission guidelines

This assignment must be coded in C. Any other language will result in 0 points. You programs will be compiled and graded on omega.uta.edu. Please make sure they compile and run on omega before submitting them. Code that does not compile on omega with:

gcc <filename> -o pf

will result in a 0.

Insufficient or unclear build instructions will be penalized. Code that does not run on omega gets zero points.

## Academic Integrity

This assignment must be 100% your own work. No code may be copied from friends, previous students, books, web pages, etc. All code submitted is automatically checked against a database of previous semester's graded assignments, current student's code and common web sources. By submitting your code on Canvas you are attesting that you have neither given nor received unauthorized assistance on this work. Code that is copied from an external source or used as inspiration, excluding the course github or blackboard, will result in a 0 for the assignment and referral to the Office of Student Conduct.