Programming Assignment #4

CSci 430 Spring 2019

Dates:

Assigned:	Monday March 5, 2019
Due:	Wednesday April 10, 2019 (before Midnight)

Objectives:

- Better understand page replacement memory management properties through implementation of page replacement schemes.
- Implement basic page replacement schemes, such as least recently used and first-in-first-out (FIFO)
- Practice C/C++ usage and implementation of queues and stacks needed for memory management algorithms.
- Be able to compare, contrast and understand the relative performance of different replacement schemes.

Description:

In programming assignment #4 you are to write a simulation of a page replacement memory management system. You will be implementing least recently used (LRU) and first-in-first-out (FIFO) page replacement schemes (Chapter 8, pg 369).

Your program should read from a file to get information about the sequence of page faults that occur. I will provide a simple example of such a text file. It consists simply of one integer value per line, which represent a sequence of page faults that occur in the simulation. For example:

```
----------- Begin Input File pagestream.txt --------
2
3
2
1
5
2
4
5
3
2
5
2
------------ End Input File -------------------------
```

represents the sequence of page address faults/requests for the Figure 8.15 example from our text book.

Your program should read 3 pieces of information from the command line: simulation file name, replacement scheme [lru or fifo], and the number of physical frames in memory. For example:

```
$ p3.exe pagestream.sim lru 3
```

To get you started, I have included a main() function and a main event loop in a file called p3-start.cpp, that can open and read in the page stream file into an array. Your main job is to implement the least recently used (LRU)and first-in-first-out FIFO page replacement algorithms. The FIFO page replacement should be the simplier of the two, so I would suggest you get that scheme working first. In FIFO, of course, when there is a page reference 'miss', you simply need to keep a pointer or index into your frames indicating the next physical frame number that should be replaced.

Your program should take the indicated stream of page addresses (a file name) as the first parameters. The second parameter indicates the total number of frames that the system is simulating (in the above example this is 3, the same as the example shown in Figure 8.15). Your implementation should be able to simulate systems with up to 10 physical frames at a maximum.

The output for your simulation should be formatted in exactly the following format. To get exact output format and correct answers for the include pagestream.sim input file, look at the results in the zip file called pagestream-lru-3.out and pagestream-fifo-3.out.

```
Time: 0
Page: 2
Frame1: 2
Frame2: 0
Frame3: 0
Hit ratio: 0 / 12 (0)


Time: 1
Page: 3
Frame1: 2
Frame2: 3
Frame3: 0
Hit ratio: 0 / 12 (0)


Time: 2
Page: 2
Frame1: 2
Frame2: 3
Frame3: 0
Hit ratio: 1 / 12 (0.0833333)

etc.
```

To interpret, for each page reference in the pagestream, you will print out the time, the page that was referenced (time starts at 0), the Page reference that was made at that time, and the state of the system frames after that page is processed.

If you use the pagestream sequence from figure 8.15 with a system frame size of 3 you should be able to reproduce the LRU and FIFO results shown. I have given you example correct outputs for both FIFO and LRU page replacement, for systems with both 3 and 4 physical frames of memory. Use these correct example outputs to test that your implementation is working correctly. I will test your program using different pagestream sequences and different system frame sizes, so make sure you thoroughly test your algorithm yourself before submitting.