Programming Assignment #3

CSci 530, Summer 2015

Dates:

Assigned:	Monday July 6, 2015
Due:	Friday July 17 (before Midnight)

Objectives:

- Better understand page replacement memory management properties through implementation of page replacement schemes.
- Implement and understand basic page replacement schemes, such as least recently used and optimal page replacement.
- 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 #3 you are to write a simulation of a page replacement memory management system. You will be implementing least recently used (LRU) and optimal (OPT) 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 opt], and the number of physical frame in memory. For example:

```
$ p2.exe pagestream.sim lru 3
```

To get you started, I have included a main() function and a main event loop in a file called p2-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 and optimal page replacement algorithms (lru() and optimal() functions shown here). They should take the indicated stream of page addresses 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 of our text).

The output for your simulation should be formatted to display exactly as follows. To get the exact output format you should use 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-opt-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 (time starts at 0), the page that was referenced, and the state of the system frames after that page is processed. Some other notes, page references are always positive integers: 1, 2, 3, etc. In the example output, we use 0 as a flag to indicate an empty frame, one that has not yet been loaded with any page.

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 OPT results shown. There is one example input and output included with the zip file for this programming assignment, to help you get started with testing. 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.