Assignment 4: Memory Management

In this assignment, you will write a program that simulates a simple virtual memory system using the three page replacements policies studied in class: First In First Out (FIFO), Optimal and Least Recently Used (LRU). Your program reads an input file named `input.txt` containing a sequence of page requests (page numbers) and generates an output file named `output.txt` that shows how the input pages are mapped into physical frames.

The first line in the input file has three integers. The first integer is the number of pages, the second integer is the number of frames, and the third integer is the number of page access requests. The remaining lines in the input file are page access requests (page numbers), with each page request appearing on a separate line. Page numbers and frame numbers are 0-based. In an interesting input, the number of frames is less than the number of pages; otherwise, the problem is trivial. Furthermore, in an interesting input, some pages are requested multiple times (the same page number appears multiple times in the sequence). Because the number of frames is typically smaller than the number of pages, the same page may be mapped to a different frame each time it is requested.

In your simulation, assume **pure demand paging**, that is, pages are loaded into physical frames only when they are requested. So, initially, no pages are loaded, and all frames are free (available). If the number of frames is x, the first x pages accessed will be trivially mapped to the x frames without having to unload (replace) any pages. When a request to access the $(x + 1)$th page arrives, your program must unload one of the x pages that have been loaded into physical frames and replace it with the new page. The selection of the page to be unloaded (replaced) depends on the page replacement policy.

Your program must implement three replacement policies: FIFO, Optimal and LRU:

- In the FIFO policy, the first page loaded into a physical frame is selected for unloading (replacement).
- In the Optimal policy, the page that will not be accessed for the longest time in the future is selected for unloading (replacement). This policy may be implemented by associating with each page-table entry a number indicating its next-use time in the future. When replacement is needed, replace the page with the greatest next-use number. If a page is not referenced in the future, set its next-use time to INFINITY. If there are multiple pages with INFINITE next-use times, replace the page that is currently in the smallest frame number.
- In the LRU policy, the page that has not been accessed for the longest time is selected for unloading (replacement). This policy may be implemented by associating with each

page-table entry a time stamp indicating the latest time at which the page was accessed. When replacement is needed, replace the page with the smallest time stamp.

The output file will have one line for each page request, indicating how that page request was handled. The last line for each algorithm reports the number of page faults. As shown in the example below, the output file format is self- explanatory.

Example

Here is an example input file and the corresponding output file. The input has 8 pages, 4 frames and 12 page requests.

Input

```
8    4     12
4
3
4
6
1
6
4
5
2
4
6
1
```

Output

```
FIFO
Page 4 loaded into Frame 0
Page 3 loaded into Frame 1
Page 4 already in Frame 0
Page 6 loaded into Frame 2
Page 1 loaded into Frame 3
Page 6 already in Frame 2
Page 4 already in Frame 0
Page 4 unloaded from Frame 0, Page 5 loaded into Frame 0
Page 3 unloaded from Frame 1, Page 2 loaded into Frame 1
Page 6 unloaded from Frame 2, Page 4 loaded into Frame 2
Page 1 unloaded from Frame 3, Page 6 loaded into Frame 3
Page 5 unloaded from Frame 0, Page 1 loaded into Frame 0
9 page faults
```

Optimal
Page 4 loaded into Frame 0
Page 3 loaded into Frame 1
Page 4 already in Frame 0
Page 6 loaded into Frame 2
Page 1 loaded into Frame 3
Page 6 already in Frame 2
Page 4 already in Frame 0
Page 3 unloaded from Frame 1, Page 5 loaded into Frame 1
Page 5 unloaded from Frame 1, Page 2 loaded into Frame 1
Page 4 already in Frame 0
Page 6 already in Frame 2
Page 1 already in Frame 3
6 page faults

LRU
Page 4 loaded into Frame 0
Page 3 loaded into Frame 1
Page 4 already in Frame 0
Page 6 loaded into Frame 2
Page 1 loaded into Frame 3
Page 6 already in Frame 2
Page 4 already in Frame 0
Page 3 unloaded from Frame 1, Page 5 loaded into Frame 1
Page 1 unloaded from Frame 3, Page 2 loaded into Frame 3
Page 4 already in Frame 0
Page 6 already in Frame 2
Page 5 unloaded from Frame 1, Page 1 loaded into Frame 1
7 page faults

Submission

Implement your code in C and name it `MemoryManagement.c`. Submit your code file on Canvas.