OS LAB-6

Mruganshi Gohel

B20CS014

Simulating page replacement algorithms

How to run?

```
g++ main.cpp -o main.exe
./main.exe
```

the main goal is to evaluate the effectiveness of each page replacement algorithm which affects the performance of memory management.

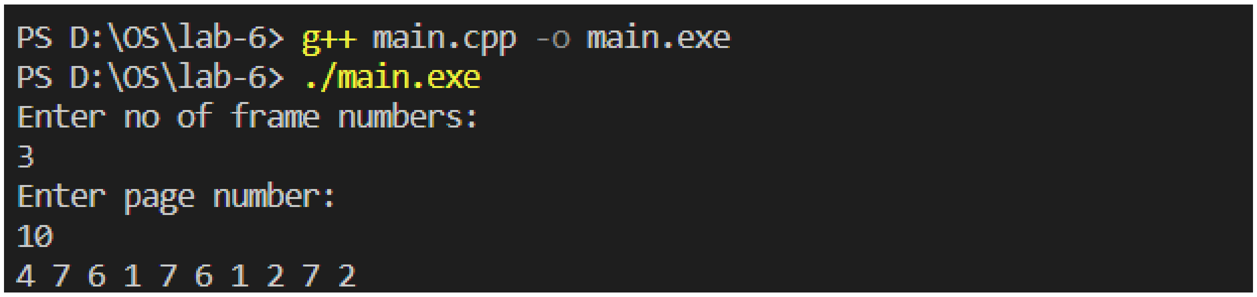
- First-in-First-out (FIFO) algorithm
- least Recently Used (LRU) algorithm
- Optimal algorithm

we will calculate the page fault rate of each page replacement algorithm for the given page reference string for one process only. the program will accept three arguments: the number of page frames of the physical memory, the number of pages, and all pages.

Input

In input, we will have

- No page frames
- reference string



```
PS D:\OS\lab-6> g++ main.cpp -o main.exe
PS D:\OS\lab-6> ./main.exe
Enter no of frame numbers:
3
Enter page number:
10
4 7 6 1 7 6 1 2 7 2
```

Algorithms implementation

1. FIFO

the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

```
1- Start traversing the pages.
 i) If set holds less pages than capacity.
   a) Insert page into the set one by one until 
      the size  of set reaches capacity or all 
      page requests are processed.
   b) Simultaneously maintain the pages in the
      queue to perform FIFO.
   c) Increment page fault
 ii) Else 
   If current page is present in set, do nothing.
   Else 
     a) Remove the first page from the queue
        as it was the first to be entered in
        the memory
     b) Replace the first page in the queue with 
        the current page in the string.
     c) Store current page in the queue.
     d) Increment page faults.

2. Return page faults.
```

2. LRU

In the least **R**ecently **U**sed (LRU) algorithm is a Greedy algorithm where the page to be replaced is the least recently used. The idea is based on the locality of reference, the least recently used page is not likely

```
Let capacity be the number of pages that
memory can hold.  Let set be the current
set of pages in memory.

1- Start traversing the pages.
 i) If set holds less pages than capacity.
   a) Insert page into the set one by one until
      the size  of set reaches capacity or all
      page requests are processed.
   b) Simultaneously maintain the recent occurred
      index of each page in a map called indexes.
   c) Increment page fault
 ii) Else
   If current page is present in set, do nothing.
   Else
     a) Find the page in the set that was least
     recently used. We find it using index array.
     We basically need to replace the page with
     minimum index.
     b) Replace the found page with current page.
     c) Increment page faults.
     d) Update index of current page.

2. Return page faults.
```

3. Optimal

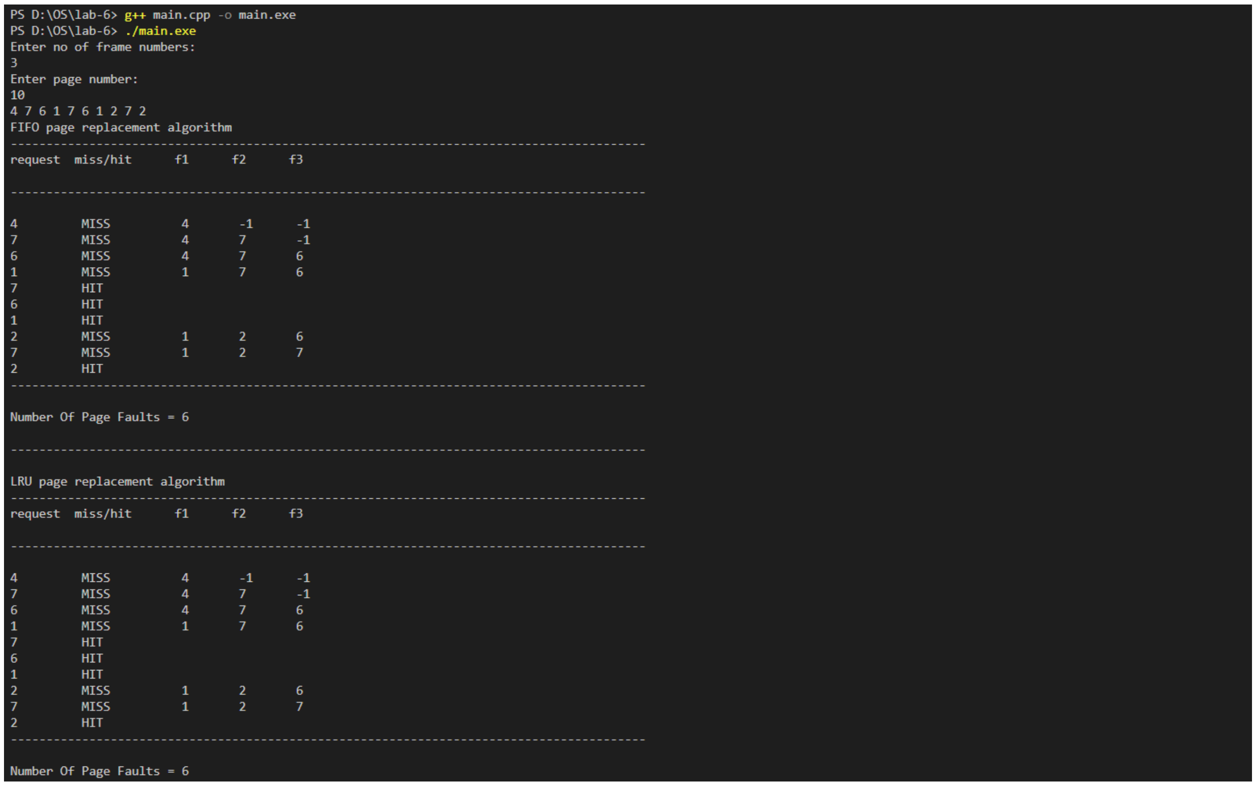
In this algorithm, OS replaces the page that will not be used for the longest period of time in the future.

The idea is simple, for every reference we do the following :

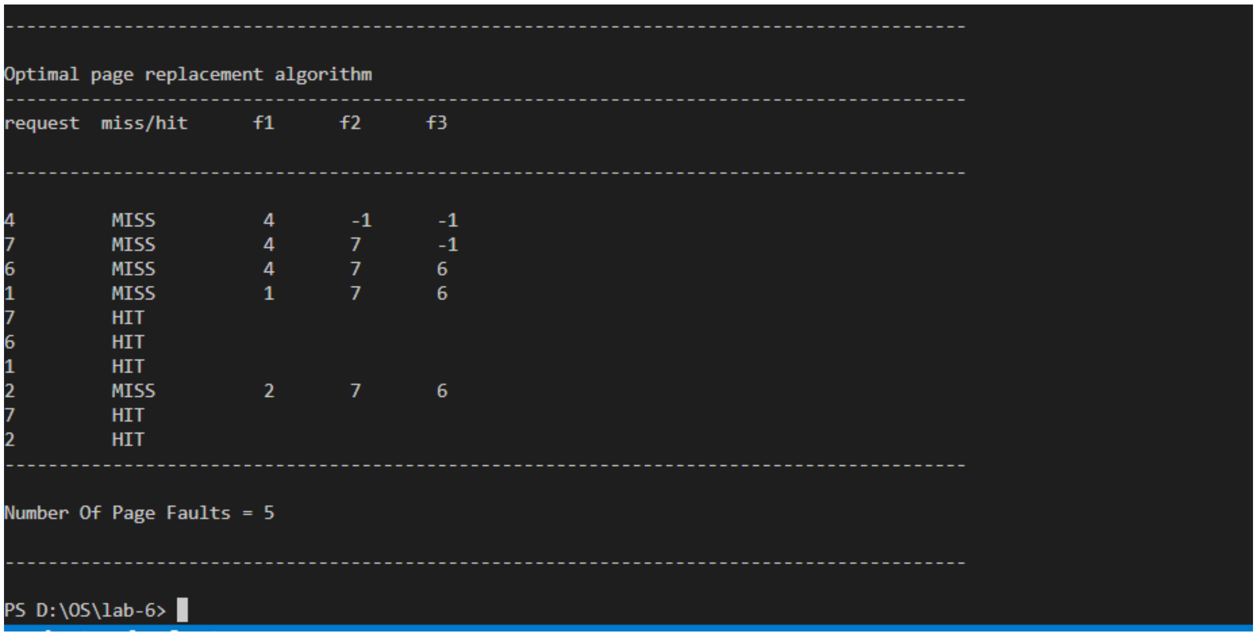
1. If the referred page is already present, increment the hit count.
2. If not present, find a page that is never referenced in the future. If such a page exists, replace this page with a new page. If no such page exists, find a page that is referenced farthest in the future. Replace this page with the new page.

Output

below is the output shown for one input and for all algorithms.



```
PS D:\OS\lab-6> g++ main.cpp -o main.exe
PS D:\OS\lab-6> ./main.exe
Enter no of frame numbers:
3
Enter page number:
10
4 7 6 1 7 6 1 2 7 2
FIFO page replacement algorithm
------------------------------------------------------------------------------------------
request  miss/hit      f1      f2      f3

------------------------------------------------------------------------------------------

4        MISS          4       -1      -1
7        MISS          4       7       -1
6        MISS          4       7       6
1        MISS          1       7       6
7        HIT
6        HIT
1        HIT
2        MISS          1       2       6
7        MISS          1       2       7
2        HIT
------------------------------------------------------------------------------------------

Number Of Page Faults = 6

------------------------------------------------------------------------------------------

LRU page replacement algorithm
------------------------------------------------------------------------------------------
request  miss/hit      f1      f2      f3

------------------------------------------------------------------------------------------

4        MISS          4       -1      -1
7        MISS          4       7       -1
6        MISS          4       7       6
1        MISS          1       7       6
7        HIT
6        HIT
1        HIT
2        MISS          1       2       6
7        MISS          1       2       7
2        HIT
------------------------------------------------------------------------------------------

Number Of Page Faults = 6
```

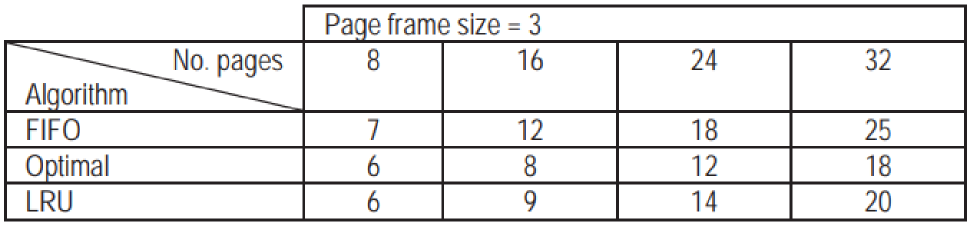


```
------------------------------------------------------------------------------------------

Optimal page replacement algorithm
------------------------------------------------------------------------------------------
request  miss/hit      f1      f2      f3

------------------------------------------------------------------------------------------

4        MISS          4       -1      -1
7        MISS          4       7       -1
6        MISS          4       7       6
1        MISS          1       7       6
7        HIT
6        HIT
1        HIT
2        MISS          2       7       6
7        HIT
2        HIT
------------------------------------------------------------------------------------------

Number Of Page Faults = 5

------------------------------------------------------------------------------------------

PS D:\OS\lab-6> █
```

Comparison

A random sequence of numbers is chosen because the application generates different sequences, and the purpose is to
test the performance of the three algorithms with the same sequence. Thus, the following sequence is chosen:
856253542353262568562342137543l5
and it is used for respectively 8 pages, 16 pages, 24 pages, and 32 pages.

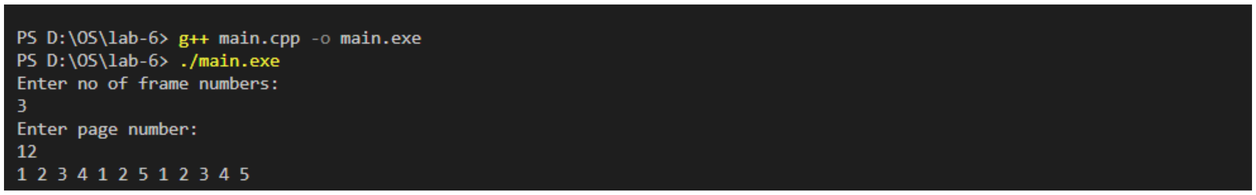
In the first step, the program requires the number of pages that are pretended to be in the main memory. After that, it requires putting the sequence of the pages. In the next step, a menu is printed. A choice can be made. The number of frames is entered, which is the capacity of the main memory. The application shows for each step the pages that are currently in main memory and the number of page faults. A table with the number of page faults, for each algorithm, with page frame size = 3 and respectively 8 pages, 16 pages, 24 pages, and 32 pages are generated.



	Page frame size = 3			
No. pages / Algorithm	8	16	24	32
FIFO	7	12	18	25
Optimal	6	8	12	18
LRU	6	9	14	20

Belady’s Anomaly condition

Assuming a system that has no pages loaded in the memory and uses the FIFO Page replacement algorithm. Consider the following reference string:



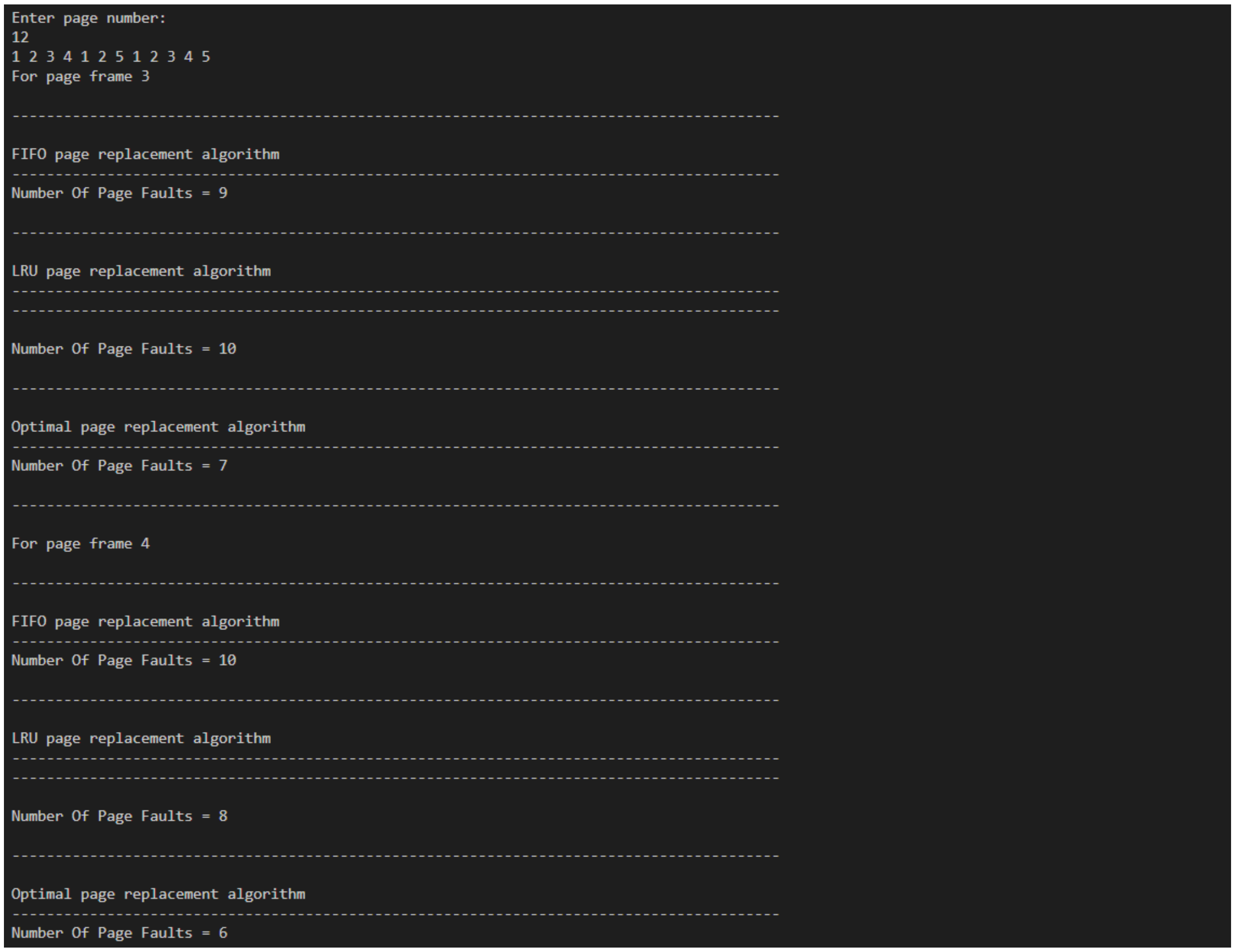
```
PS D:\OS\lab-6> g++ main.cpp -o main.exe
PS D:\OS\lab-6> ./main.exe
Enter no of frame numbers:
3
Enter page number:
12
1 2 3 4 1 2 5 1 2 3 4 5
```

I have taken page frame numbers from 3 to 8 , and by below analysis and my code output I was able to observe belady's Anomaly in frame 4.

Analysis :

Case-1: If the system has 3 frames, the given reference string the using FIFO page replacement algorithm yields a total of 9-page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

Case-2: If the system has 4 frames, the given reference string using the FIFO page replacement algorithm yields a total of 10-page faults. The diagram below illustrates the pattern of the page faults occurring in the example.



```
Enter page number:
12
1 2 3 4 1 2 5 1 2 3 4 5
For page frame 3

-----------------------------------------------------------------------------------------

FIFO page replacement algorithm
-----------------------------------------------------------------------------------------
Number Of Page Faults = 9

-----------------------------------------------------------------------------------------

LRU page replacement algorithm
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------

Number Of Page Faults = 10

-----------------------------------------------------------------------------------------

Optimal page replacement algorithm
-----------------------------------------------------------------------------------------
Number Of Page Faults = 7

-----------------------------------------------------------------------------------------

For page frame 4

-----------------------------------------------------------------------------------------

FIFO page replacement algorithm
-----------------------------------------------------------------------------------------
Number Of Page Faults = 10

-----------------------------------------------------------------------------------------

LRU page replacement algorithm
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------

Number Of Page Faults = 8

-----------------------------------------------------------------------------------------

Optimal page replacement algorithm
-----------------------------------------------------------------------------------------
Number Of Page Faults = 6
```

It can be seen from the above example that on increasing the number of frames (in frame 4) while using the FIFO page replacement algorithm, the number of **page faults increased** from 9 to 10.

It is not necessary that every string reference pattern cause a Belady anomaly in FIFO but there is certain kind of string references that worsen the FIFO performance by increasing the number of frames.