Name: ***Kumar Deshmukh***
Roll: ***33143***
Batch: ***G9***

Problem Statement:

Implement the C program for Page Replacement Algorithms: FCFS, LRU, and Optimal for frame size as minimum three.

Theory:

Page replacement

- The page replacement algorithm decides which memory page is to be replaces.
- The process of replacement is sometimes called swap out or write to disk
- Page replacement is done when the requested page is not found in the main memory (page fault).
- There are two main aspects of virtual memory, Frame Allocation and Page Replacement.
- It is very important to have optimal frame allocation and page replacement algorithm.
- Frame allocation is all about how many are to be allocated to the process while the page replacement is all about determining the page number which needs to be replaced in order to make space for the requested page.

Page fault

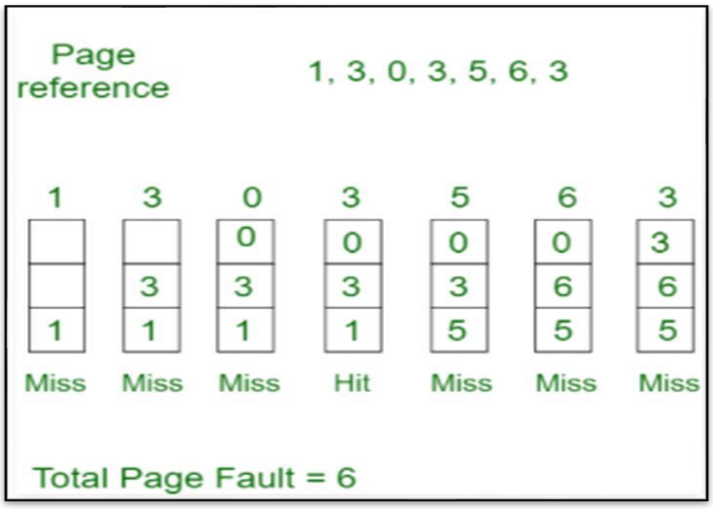
- Page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.
- Since actual physical memory is much smaller than virtual memory, page faults happen.
- In case of page fault, Operating System might have to replace one of the existing pages with newly added page.
- Different page replacement algorithms suggest one of the existing pages which page to replace.
- The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms

1) First In First Out (FIFO) Page Replacement

- This is the simplest page replacement algorithm
- In this algorithm, the operating system keeps track of all pages in the memory in a queue.
- When a page needs to be replaced page in the front of the queue is selected for removal

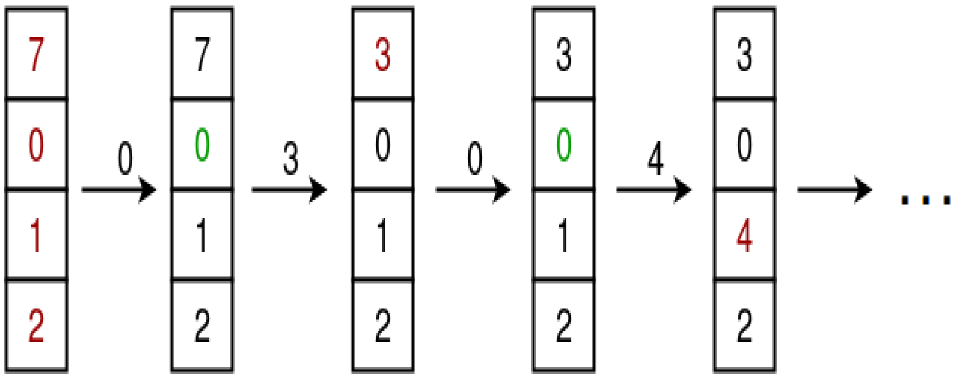
Example:



- Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots → 3 Page Faults.
- When 3 comes, it is already in memory so → 0 Page Faults.
- Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e., 1 → 1 Page Fault.
- 6 comes, it is also not available in memory so it replaces the oldest page slot i.e., 3 → 1 Page Fault.
- Finally, when 3 come it is not available so it replaces 0 → 1 page fault

2) Least Recently Used(LRU) Page Replacement

- Least Recently Used (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used. The idea is based on locality of reference, the least recently used page is not likely
  Example:

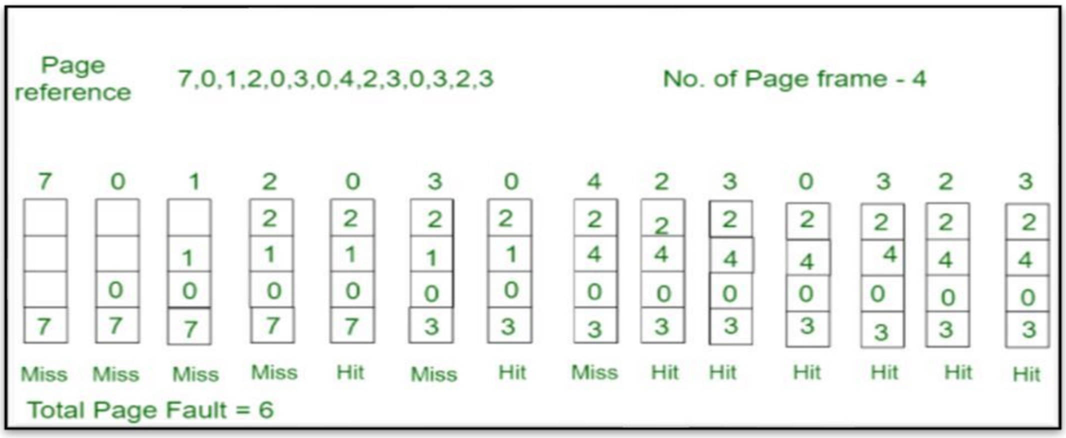


Total Page faults = 6

- Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially we have 4 page slots empty.
- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults
- 0 is already their so —> 0 Page fault.
- when 3 came it will take the place of 7 because it is least recently used —>1 Page fault
- 0 is already in memory so —> 0 Page fault.
- 4 will takes place of 1 —> 1 Page Fault
- Now for the further page reference string —> 0 Page fault because they are already available in the memory.

3) *Optimal Page Replacement*

- Those pages are replaced which would not be used for the longest duration of time in the future.
- Example:



- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots → 4 Page faults
- 0 is already there so→ 0 Page fault.
- When 3 came it will take the place of 7 because it is not used for the longest duration of time in the future. →1 Page fault.
- 0 is already there so → 0 Page fault.
- 4 will take place of 1 → Page Fault.
- Now for the further page reference string → 0 Page fault because they are already available in the memory.
- Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests.
- The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analysed against it.

CODE: (.cpp) file attached

OUTPUT:

```
(base) kumar@pop-os:~/Desktop/OS/Assignments$ g++ Assig6.cpp
(base) kumar@pop-os:~/Desktop/OS/Assignments$ ./a.out

MENU
        [1] FIFO scheme
        [2] LRU scheme
        [3] Optimal scheme
        [4] EXIT
Choice: 1
Enter no. of Frames: 3
Enter no. of References: 12
Enter Frame Reference Sequence:
        Reference 1: 1
        Reference 2: 2
        Reference 3: 3
        Reference 4: 4
        Reference 5: 1
        Reference 6: 2
        Reference 7: 5
        Reference 8: 1
        Reference 9: 2
        Reference 10: 3
        Reference 11: 4
        Reference 12: 5

For FIFO policy::
1
1 2
1 2 3
2 3 4
3 4 1
4 1 2
1 2 5
2 5 3
5 3 4

Page Faults = 9
Hit Rate = 25%
Fault Rate = 75%

MENU
        [1] FIFO scheme
        [2] LRU scheme
        [3] Optimal scheme
        [4] EXIT
Choice: 1
Enter no. of Frames: 4
Enter no. of References: 12
Enter Frame Reference Sequence:
        Reference 1: 1
        Reference 2: 2
        Reference 3: 3
        Reference 4: 4
        Reference 5: 1
        Reference 6: 2
        Reference 7: 5
        Reference 8: 1
        Reference 9: 2
        Reference 10: 3
        Reference 11: 4
        Reference 12: 5
```

```
For FIFO policy::
1
1 2
1 2 3
1 2 3 4
2 3 4 5
3 4 5 1
4 5 1 2
5 1 2 3
1 2 3 4
2 3 4 5

Page Faults = 10
Hit Rate = 16.6667%
Fault Rate = 83.3333%

MENU
        [1] FIFO scheme
        [2] LRU scheme
        [3] Optimal scheme
        [4] EXIT
Choice: 2
Enter no. of Frames: 3
Enter no. of References: 12
Enter Frame Reference Sequence:
        Reference 1: 1
        Reference 2: 2
        Reference 3: 3
        Reference 4: 4
        Reference 5: 1
        Reference 6: 2
        Reference 7: 5
        Reference 8: 1
        Reference 9: 2
        Reference 10: 3
        Reference 11: 4
        Reference 12: 5

For LRU policy::
1
1 2
1 2 3
2 3 4
3 4 1
4 1 2
1 2 5
2 5 1
5 1 2
1 2 3
2 3 4
3 4 5

Page Faults = 10
Hit Rate = 16.6667%
Fault Rate = 83.3333%

MENU
        [1] FIFO scheme
        [2] LRU scheme
        [3] Optimal scheme
        [4] EXIT
Choice: 2
Enter no. of Frames: 4
Enter no. of References: 13
Enter Frame Reference Sequence:
        Reference 1: 5
        Reference 2: 7
        Reference 3: 0
        Reference 4: 1
        Reference 5: 7
        Reference 6: 6
        Reference 7: 7
        Reference 8: 2
        Reference 9: 1
        Reference 10: 6
        Reference 11: 7
        Reference 12: 6
        Reference 13: 1

For LRU policy::
5
5 7
5 7 0
5 7 0 1
5 0 1 7
0 1 7 6
0 1 6 7
1 6 7 2
6 7 2 1
7 2 1 6
2 1 6 7
2 1 7 6
2 7 6 1

Page Faults = 6
Hit Rate = 53.8462%
Fault Rate = 46.1538%

MENU
        [1] FIFO scheme
        [2] LRU scheme
        [3] Optimal scheme
        [4] EXIT
Choice: 3
Enter no. of Frames: 4
Enter no. of References: 13
Enter Frame Reference Sequence:
        Reference 1: 5
        Reference 2: 7
        Reference 3: 0
        Reference 4: 1
        Reference 5: 7
        Reference 6: 6
        Reference 7: 7
        Reference 8: 2
        Reference 9: 1
        Reference 10: 6
        Reference 11: 7
        Reference 12: 6
        Reference 13: 1

For Optimal policy::
5
5 7
5 7 0
5 7 0 1
5 7 0 1
5 0 1 6
0 1 6 7
0 1 6 2
0 1 6 2
0 1 6 2
0 6 2 7
0 6 2 7
```

```
0 6 2 1

Page Faults = 9
Hit Rate = 30.7692 %
Fault Rate = 69.2308 %

MENU
        [1] FIFO scheme
        [2] LRU scheme
        [3] Optimal scheme
        [4] EXIT
Choice: 4

Exited Program
(base) kumar@pop-os:~/Desktop/OS/Assignments$
```

Conclusion:

We implemented FIFO, LRU & Optimal Page Replacement policies