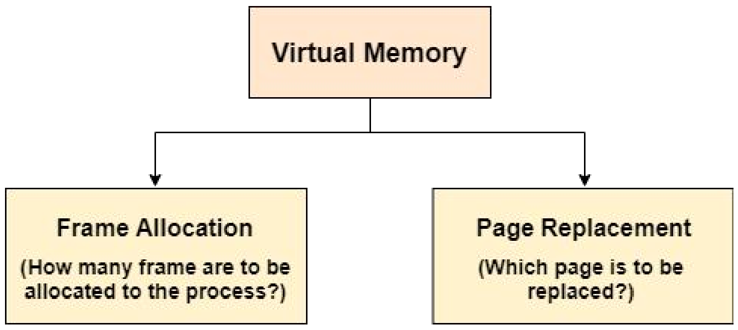
Assignment No. 7 Write a program to simulate Page replacement algorithm.
Theory:

Virtual memory uses both hardware and software to enable a computer to compensate for main memory shortages, temporarily transferring data from random access memory (RAM) to disk storage. Computers have a finite amount of RAM, so memory will eventually run out when multiple programs run at the same time. A system using virtual memory uses a section of the hard drive to emulate RAM. With virtual memory, a system can load larger or multiple programs running at the same time, enabling each one to operate as if it has more space, without having to purchase more RAM.


Virtual Memory
Frame Allocation
(How many frame are to be allocated to the process?)
Page Replacement
(Which page is to be replaced?)

There are two main aspects of virtual memory, Frame allocation and Page Replacement. It is very important to have the optimal frame allocation and page replacement algorithm. Frame allocation is all about how many frames 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 replacement algorithms are an important part of virtual memory management and it helps the OS to decide which memory page can be moved out making space for the currently needed page. However, the ultimate objective of all page replacement algorithms is to reduce the number of page faults.

FIFO - This is the simplest page replacement algorithm. In this algorithm, 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.

LRU - In this algorithm page will be replaced which is least recently used.

OPTIMAL- In this algorithm, pages are replaced which would not be used for the longest duration of time in the future. This algorithm will give us less page fault when compared to other page replacement algorithms.

Algorithm for FIFO:

```
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.
```

Algorithm for LRU:

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.
```

Algorithm for Optimal:

1. Push the first page in the memory as per the memory demand.
2. Push the second page as per the memory demand.
3. Push the third page until the memory is full.
4. As the queue is full, the page which is least recently used is popped.
5. Repeat step 4 until the page demand continues and until the processing is over.
6. Terminate the program.

Conclusion: Hence we have successfully simulated Page Replacement Algorithms.