EC-231

Operating Systems LAB

PROJECT

Submitted by

Name:

Abdullah Javed (20-CE-035)

Shiza Iftikhar (20-CE-019)

Fatima Asim (20-CE-029)

Department of Computer Engineering

HITEC University Taxila

Lab. Instructor

Kaynat Rana

Project Experiment

Page Replacement

Objective

Purpose of the project is to familiarize the students with Page Replacement Algorithm.

Software Tools

- https://www.onlinegdb.com/online_c++_compiler

Theory

Introduction to Paging

Paging is a type of memory management scheme which the computer uses to store and receive data from the secondary storage and uses that in the main memory. This scheme allows the physical address space of a process to be non-contiguous.

e.g.

There is a process of size: 4

Bytes The size of a single page: 2

Bytes Number of pages/Process: 4/2 = 2

To follow the concept of paging the MMU makes a page table. Every process has its own page table. The number of pages in a page table is the same as the number of pages in a process.

E.g.

The process 1 above, will have two entries in its page table. The page table however contains the "Frame No" which is the address in main memory where the data required is stored.

Page Replacement Algorithms

In the Operating systems that use the concept of paging, page replacement algorithms are required to choose the page that needs to be replaced when a new page comes in. When a new page is referred and it is a not present in the memory a page fault occurs and the operating systems replaces an existing page with the newer needed page. Different such algorithms suggest different ways to decide which page should be replaced, the aim of these algorithms is to reduce the number of page faults. Page faults can be described as a type of interrupt which is raised by the hardware when running a program access a memory page that is mapped into the virtual address space but is not loaded into the physical memory.

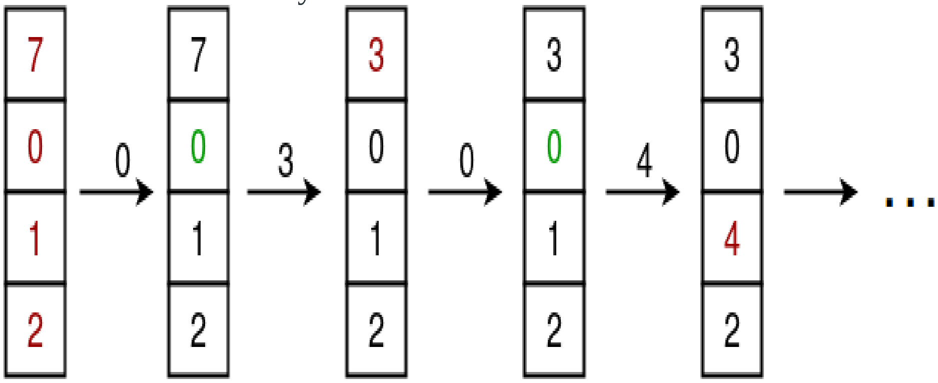
First in First Out

This page replacement algorithm works on the principle that the oldest page present in the physical memory is to be replaced first. It is the simplest of the page replacement algorithms and has a relatively low overhead. In order to keep track of the sequence of page arrivals, we can simply maintain a FIFO queue of a certain number of page frames where the pages will be kept. A page will be added at the tail of the queue and replacement will take place at the head. Therefore, the FIFO page replacement algorithm associates the time of addition into the memory with each page.

Least Recently Used (LRU)

In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

In **L**east **R**ecently **U**sed (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
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.



Total Page faults = 6

Least Frequently Used (LFU):

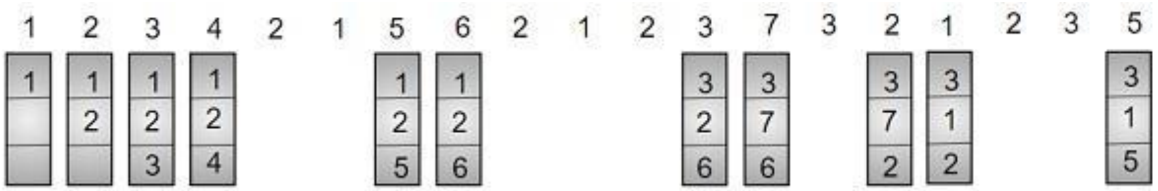
Least Frequently Used (LFU) is a caching algorithm in which the least frequently used cache block is removed whenever the cache is overflowed. In LFU we check the old page as well as the frequency of that page and if the frequency of the page is larger than the old page we cannot remove it and if all the old pages are having same frequency then take last i.e FIFO method for that and remove that page.
Min-heap data structure is a good option to implement this algorithm, as it handles insertion, deletion, and update in logarithmic time complexity. A tie can be resolved by removing the least recently used cache block. The following two containers have been used to solve the problem:

- A vector of integer pairs has been used to represent the cache, where each pair consists of the block number and the number of times it has been used. The vector is ordered in the form of a min-heap, which allows us to access the least frequently used block in constant time.
- A hashmap has been used to store the indices of the cache blocks which allows searching in constant time.

Optimal Page Replacement:

Optimal page replacement algorithm is a page replacement algorithm. A page replacement algorithm is an algorithm which decides which memory page is to be replaced. In Optimal page replacement we replace the page which is not referred to the near future, although it can't be practically implemented, but this is most optimal and have minimal miss, and is most optimal.

Let's understand by using an example and explaining it diagrammatically.



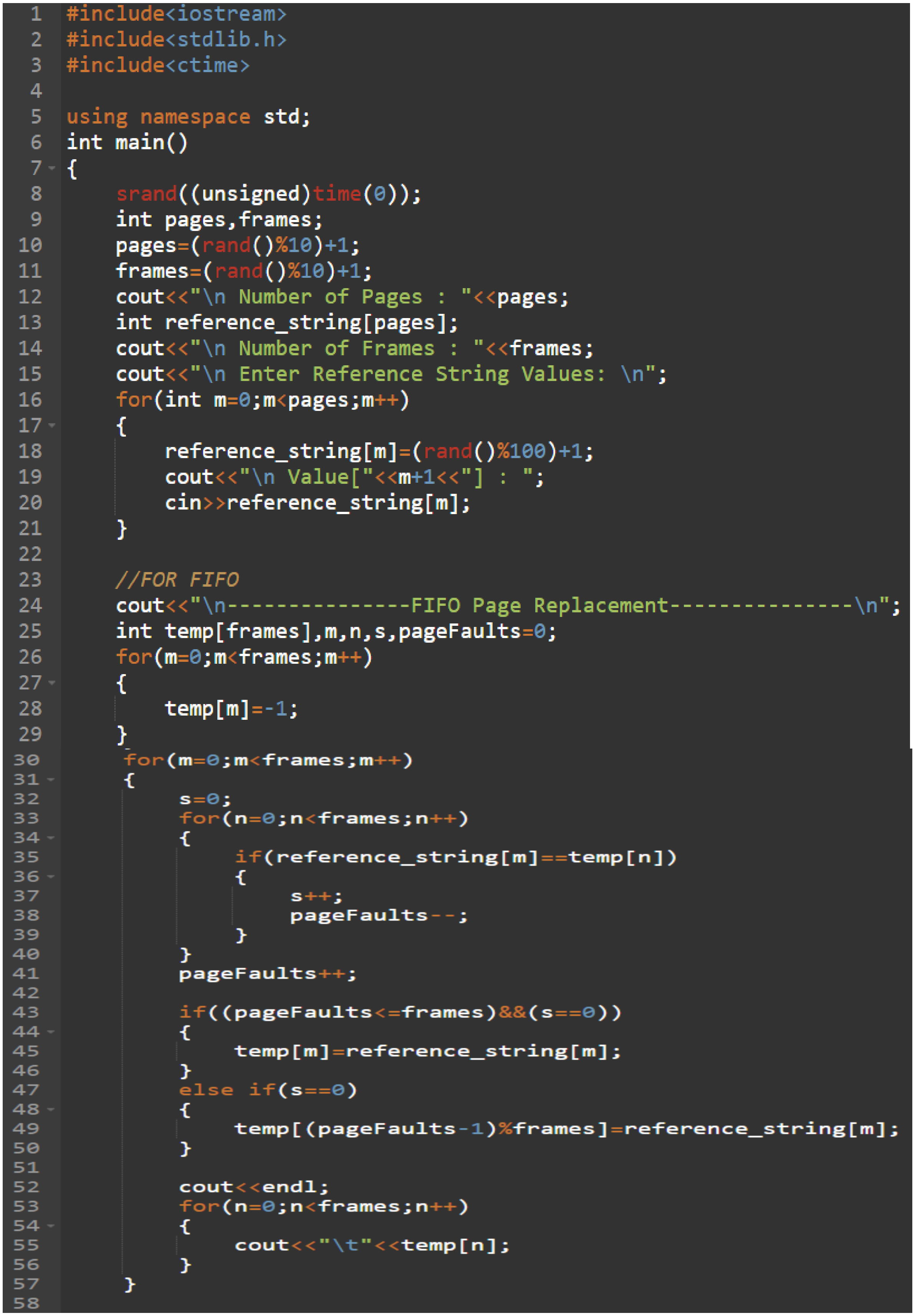
Here after allocating 1, 2 and 3 now the memory is full, so for inserting 4 we will look for the page which is not again referred in near future from 1, 2 and 3 so page 3 is not in near future so we replace that page with new page 4, and so on we will repeat the steps till we reach the end.

Example

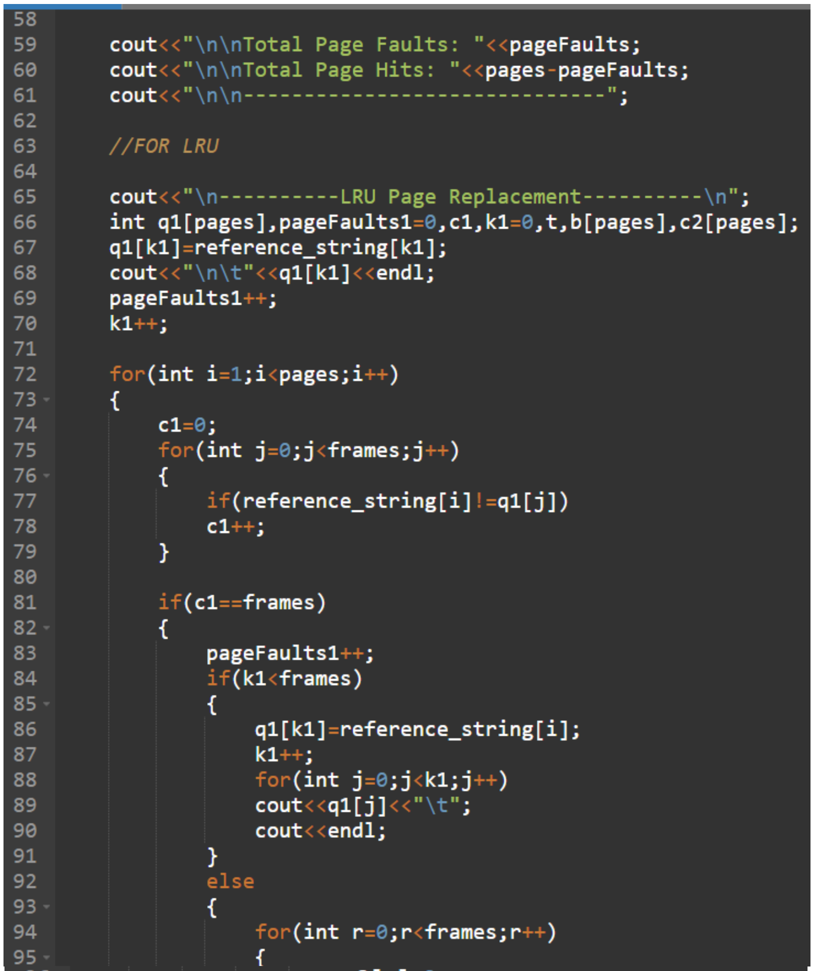
```
Input: page[] = { 1, 7, 8, 3, 0, 2, 0, 3, 5, 4, 0, 6, 1 }
   fn=3
Output: Hits = 3
   Misses = 10

Input: page[] = { 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 }
   fn = 4
Output: Hits = 7
   Misses= 6
```

Project Task:

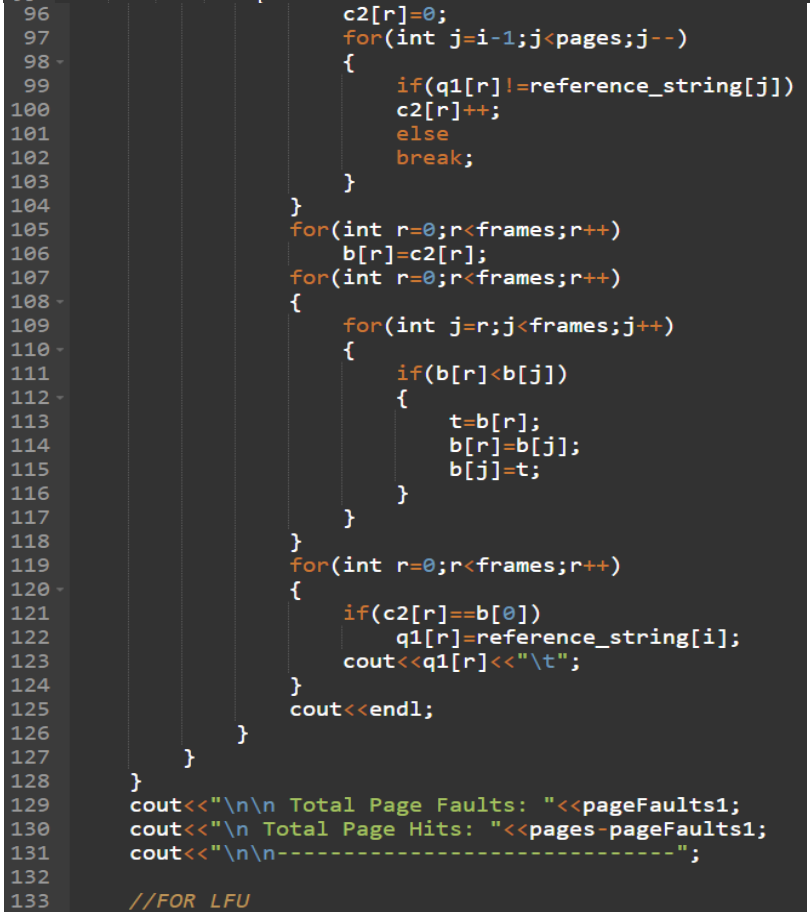
Code:



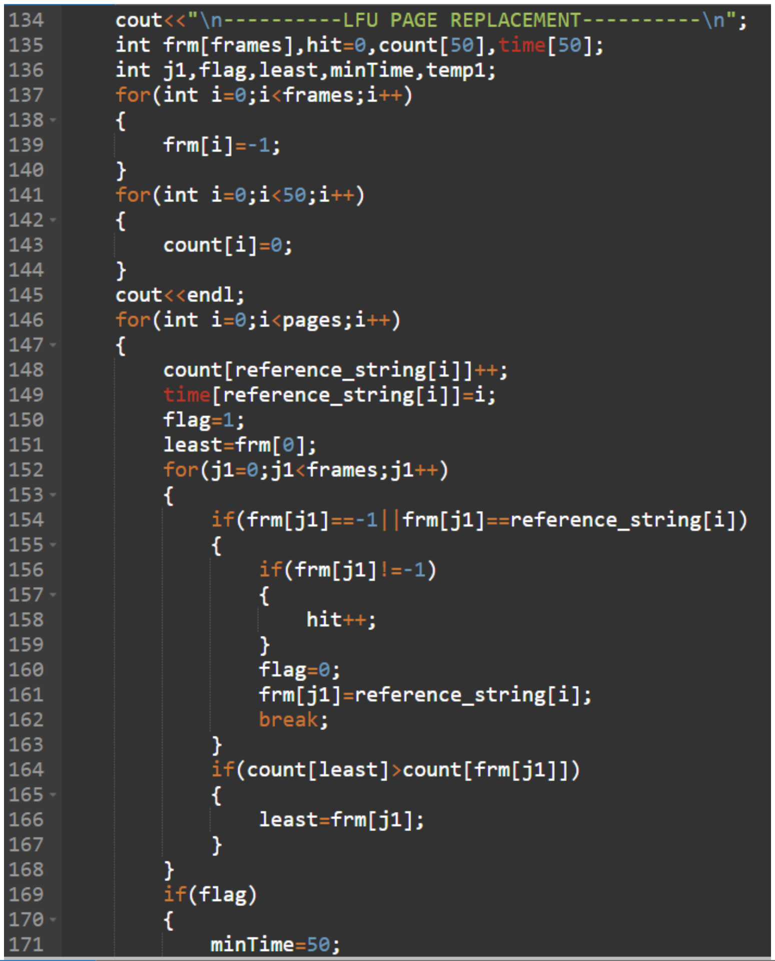
```cpp
#include<iostream>
#include<stdlib.h>
#include<ctime>

using namespace std;
int main()
{
    srand((unsigned)time(0));
    int pages,frames;
    pages=(rand()%10)+1;
    frames=(rand()%10)+1;
    cout<<"\n Number of Pages : "<<pages;
    int reference_string[pages];
    cout<<"\n Number of Frames : "<<frames;
    cout<<"\n Enter Reference String Values: \n";
    for(int m=0;m<pages;m++)
    {
        reference_string[m]=(rand()%100)+1;
        cout<<"\n Value["<<m+1<<"] : ";
        cin>>reference_string[m];
    }

    //FOR FIFO
    cout<<"\n---------------FIFO Page Replacement---------------\n";
    int temp[frames],m,n,s,pageFaults=0;
    for(m=0;m<frames;m++)
    {
        temp[m]=-1;
    }
    for(m=0;m<frames;m++)
    {
        s=0;
        for(n=0;n<frames;n++)
        {
            if(reference_string[m]==temp[n])
            {
                s++;
                pageFaults--;
            }
        }
        pageFaults++;

        if((pageFaults<=frames)&&(s==0))
        {
            temp[m]=reference_string[m];
        }
        else if(s==0)
        {
            temp[(pageFaults-1)%frames]=reference_string[m];
        }

        cout<<endl;
        for(n=0;n<frames;n++)
        {
            cout<<"\t"<<temp[n];
        }
    }

```



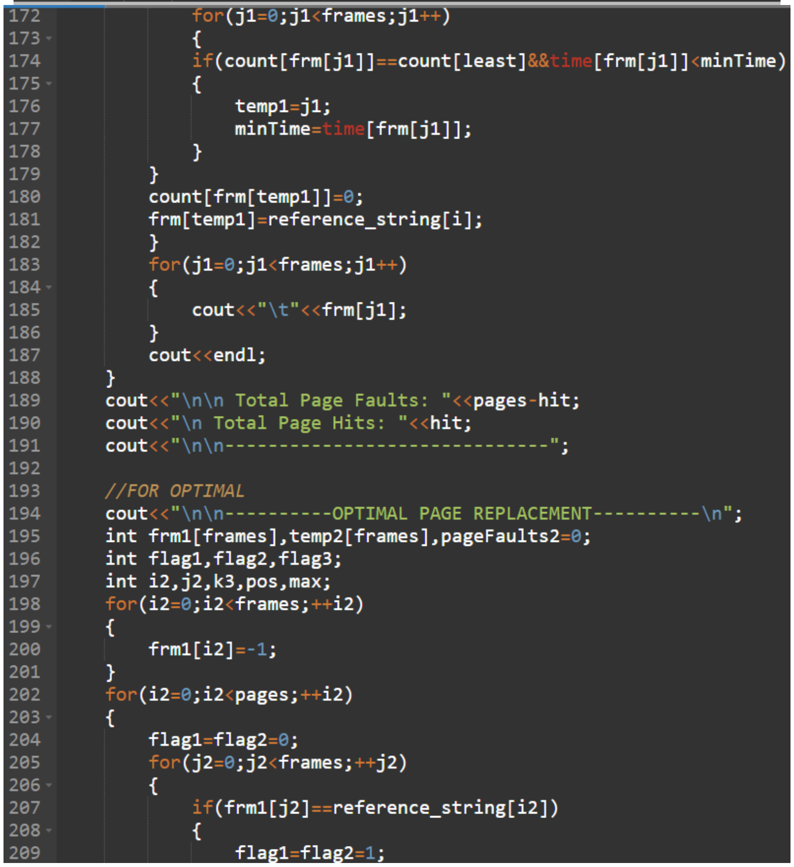
```
    cout<<"\n\nTotal Page Faults: "<<pageFaults;
    cout<<"\n\nTotal Page Hits: "<<pages-pageFaults;
    cout<<"\n\n-----------------------------";

    //FOR LRU

    cout<<"\n----------LRU Page Replacement----------\n";
    int q1[pages],pageFaults1=0,c1,k1=0,t,b[pages],c2[pages];
    q1[k1]=reference_string[k1];
    cout<<"\n\t"<<q1[k1]<<endl;
    pageFaults1++;
    k1++;

    for(int i=1;i<pages;i++)
    {
        c1=0;
        for(int j=0;j<frames;j++)
        {
            if(reference_string[i]!=q1[j])
            c1++;
        }

        if(c1==frames)
        {
            pageFaults1++;
            if(k1<frames)
            {
                q1[k1]=reference_string[i];
                k1++;
                for(int j=0;j<k1;j++)
                cout<<q1[j]<<"\t";
                cout<<endl;
            }
            else
            {
                for(int r=0;r<frames;r++)
                {
```



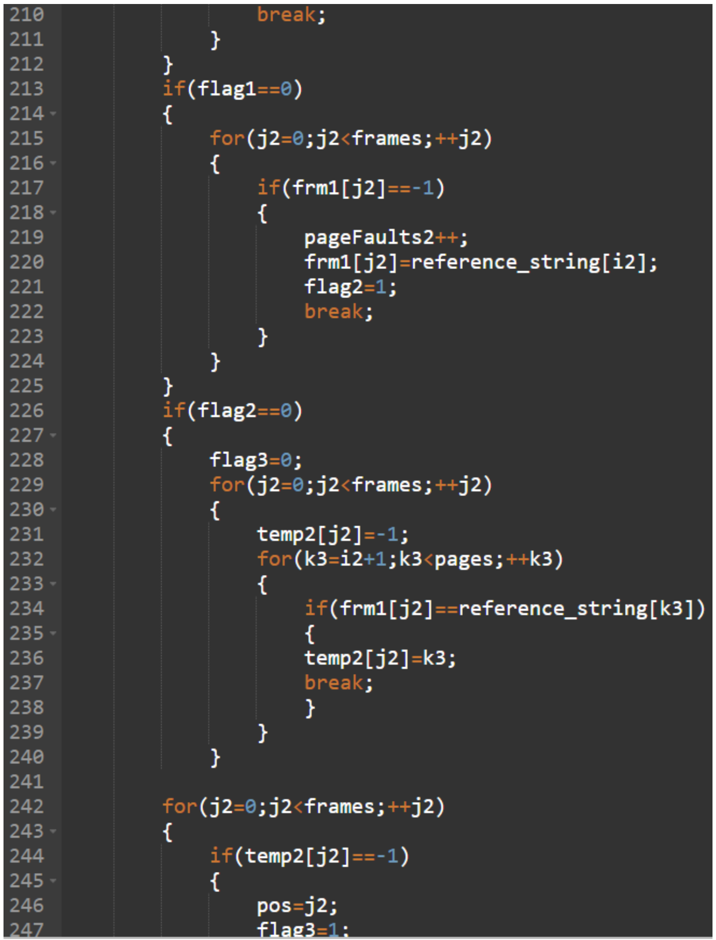
```
                    c2[r]=0;
                    for(int j=i-1;j<pages;j--)
                    {
                        if(q1[r]!=reference_string[j])
                        c2[r]++;
                        else
                        break;
                    }
                }
                for(int r=0;r<frames;r++)
                    b[r]=c2[r];
                for(int r=0;r<frames;r++)
                {
                    for(int j=r;j<frames;j++)
                    {
                        if(b[r]<b[j])
                        {
                            t=b[r];
                            b[r]=b[j];
                            b[j]=t;
                        }
                    }
                }
                for(int r=0;r<frames;r++)
                {
                    if(c2[r]==b[0])
                        q1[r]=reference_string[i];
                    cout<<q1[r]<<"\t";
                }
                cout<<endl;
            }
        }
    }
    cout<<"\n\n Total Page Faults: "<<pageFaults1;
    cout<<"\n Total Page Hits: "<<pages-pageFaults1;
    cout<<"\n\n------------------------------";

    //FOR LFU
```



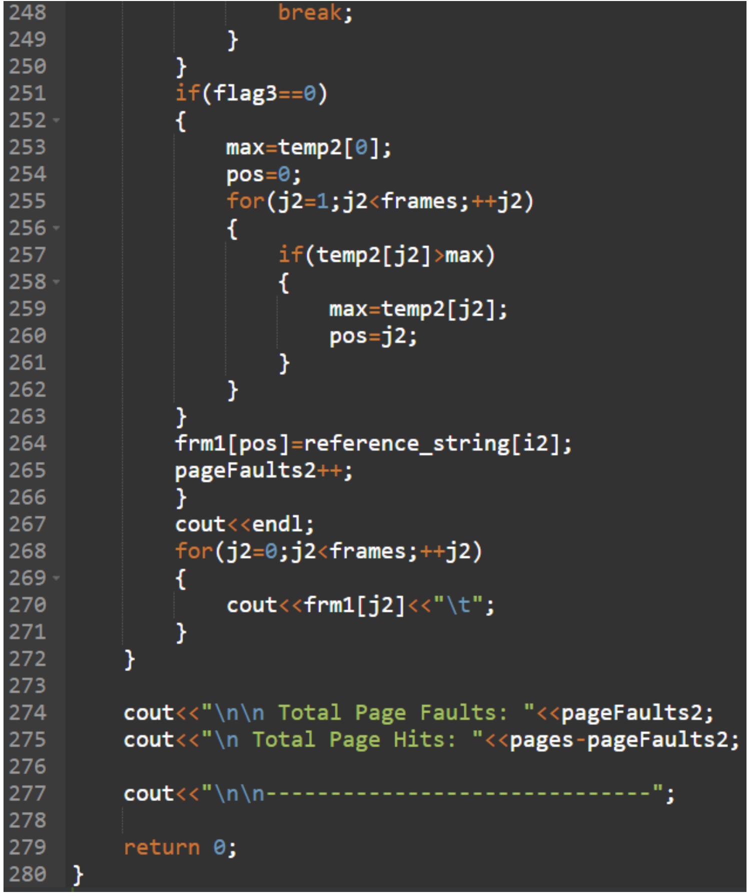
```
    cout<<"\n----------LFU PAGE REPLACEMENT----------\n";
    int frm[frames],hit=0,count[50],time[50];
    int j1,flag,least,minTime,temp1;
    for(int i=0;i<frames;i++)
    {
        frm[i]=-1;
    }
    for(int i=0;i<50;i++)
    {
        count[i]=0;
    }
    cout<<endl;
    for(int i=0;i<pages;i++)
    {
        count[reference_string[i]]++;
        time[reference_string[i]]=i;
        flag=1;
        least=frm[0];
        for(j1=0;j1<frames;j1++)
        {
            if(frm[j1]==-1||frm[j1]==reference_string[i])
            {
                if(frm[j1]!=-1)
                {
                    hit++;
                }
                flag=0;
                frm[j1]=reference_string[i];
                break;
            }
            if(count[least]>count[frm[j1]])
            {
                least=frm[j1];
            }
        }
        if(flag)
        {
            minTime=50;
```



```
            for(j1=0;j1<frames;j1++)
            {
            if(count[frm[j1]]==count[least]&&time[frm[j1]]<minTime)
            {
                temp1=j1;
                minTime=time[frm[j1]];
            }
        }
        count[frm[temp1]]=0;
        frm[temp1]=reference_string[i];
        }
        for(j1=0;j1<frames;j1++)
        {
            cout<<"\t"<<frm[j1];
        }
        cout<<endl;
    }
    cout<<"\n\n Total Page Faults: "<<pages-hit;
    cout<<"\n Total Page Hits: "<<hit;
    cout<<"\n\n------------------------------";

    //FOR OPTIMAL
    cout<<"\n\n----------OPTIMAL PAGE REPLACEMENT----------\n";
    int frm1[frames],temp2[frames],pageFaults2=0;
    int flag1,flag2,flag3;
    int i2,j2,k3,pos,max;
    for(i2=0;i2<frames;++i2)
    {
        frm1[i2]=-1;
    }
    for(i2=0;i2<pages;++i2)
    {
        flag1=flag2=0;
        for(j2=0;j2<frames;++j2)
        {
            if(frm1[j2]==reference_string[i2])
            {
                flag1=flag2=1;
```

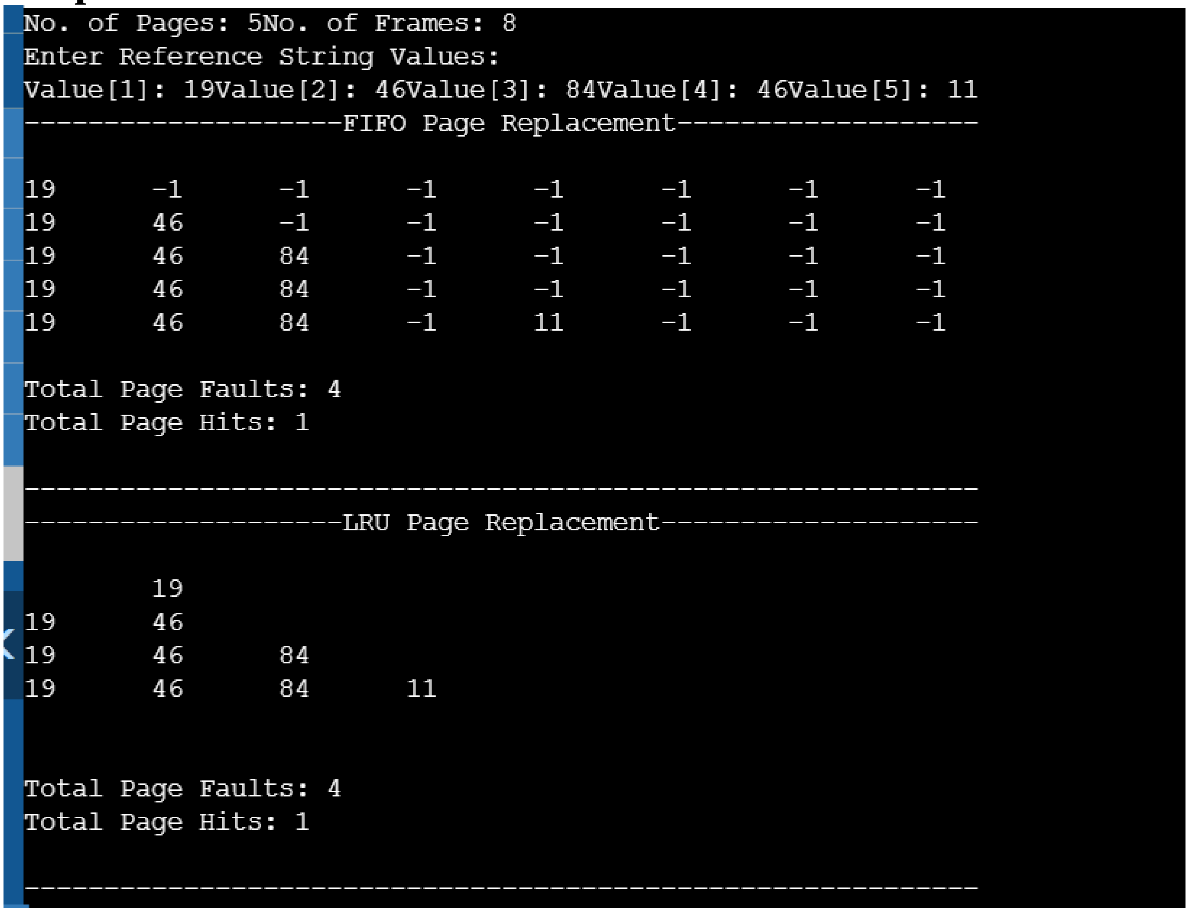


```
                break;
            }
        }
        if(flag1==0)
        {
            for(j2=0;j2<frames;++j2)
            {
                if(frm1[j2]==-1)
                {
                    pageFaults2++;
                    frm1[j2]=reference_string[i2];
                    flag2=1;
                    break;
                }
            }
        }
        if(flag2==0)
        {
            flag3=0;
            for(j2=0;j2<frames;++j2)
            {
                temp2[j2]=-1;
                for(k3=i2+1;k3<pages;++k3)
                {
                    if(frm1[j2]==reference_string[k3])
                    {
                    temp2[j2]=k3;
                    break;
                    }
                }
            }

        for(j2=0;j2<frames;++j2)
        {
            if(temp2[j2]==-1)
            {
                pos=j2;
                flag3=1;
```

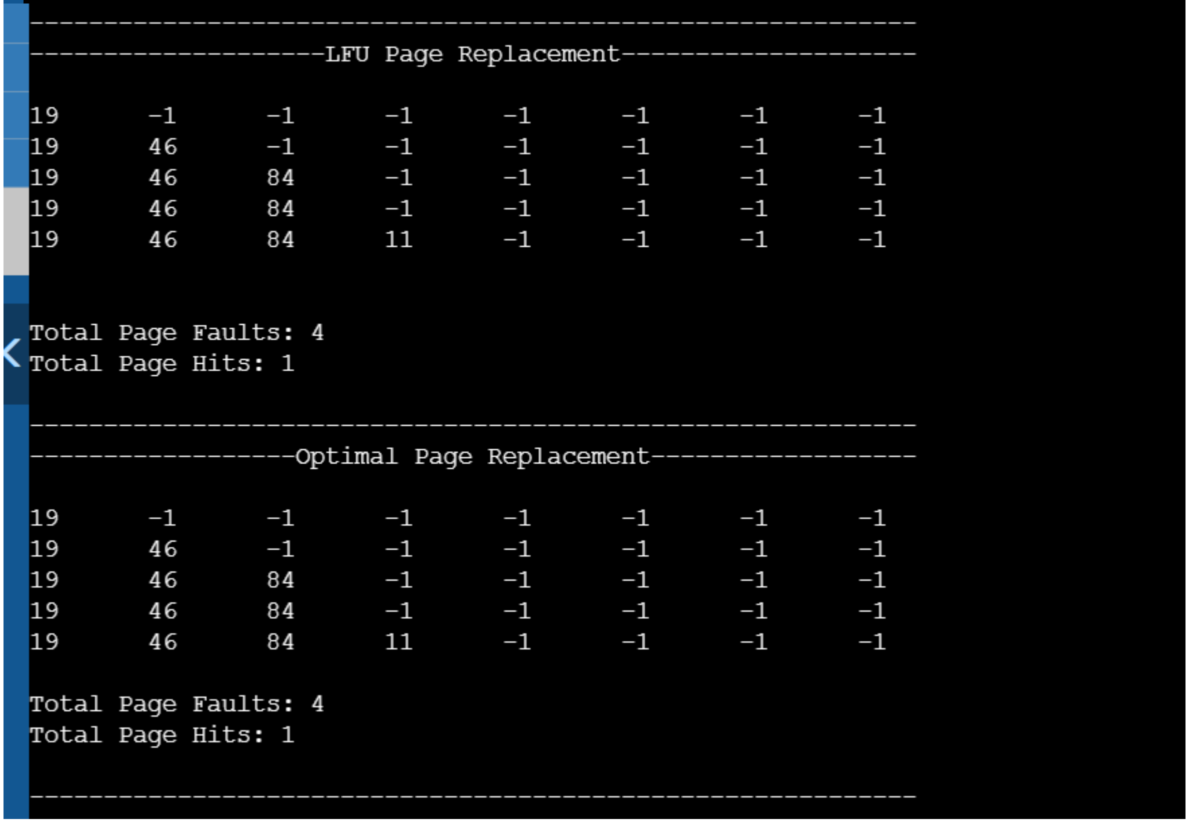


```
                break;
            }
        }
        if(flag3==0)
        {
            max=temp2[0];
            pos=0;
            for(j2=1;j2<frames;++j2)
            {
                if(temp2[j2]>max)
                {
                    max=temp2[j2];
                    pos=j2;
                }
            }
        }
        frm1[pos]=reference_string[i2];
        pageFaults2++;
        }
        cout<<endl;
        for(j2=0;j2<frames;++j2)
        {
            cout<<frm1[j2]<<"\t";
        }
    }

    cout<<"\n\n Total Page Faults: "<<pageFaults2;
    cout<<"\n Total Page Hits: "<<pages-pageFaults2;

    cout<<"\n\n------------------------------";

    return 0;
}
```

Output:

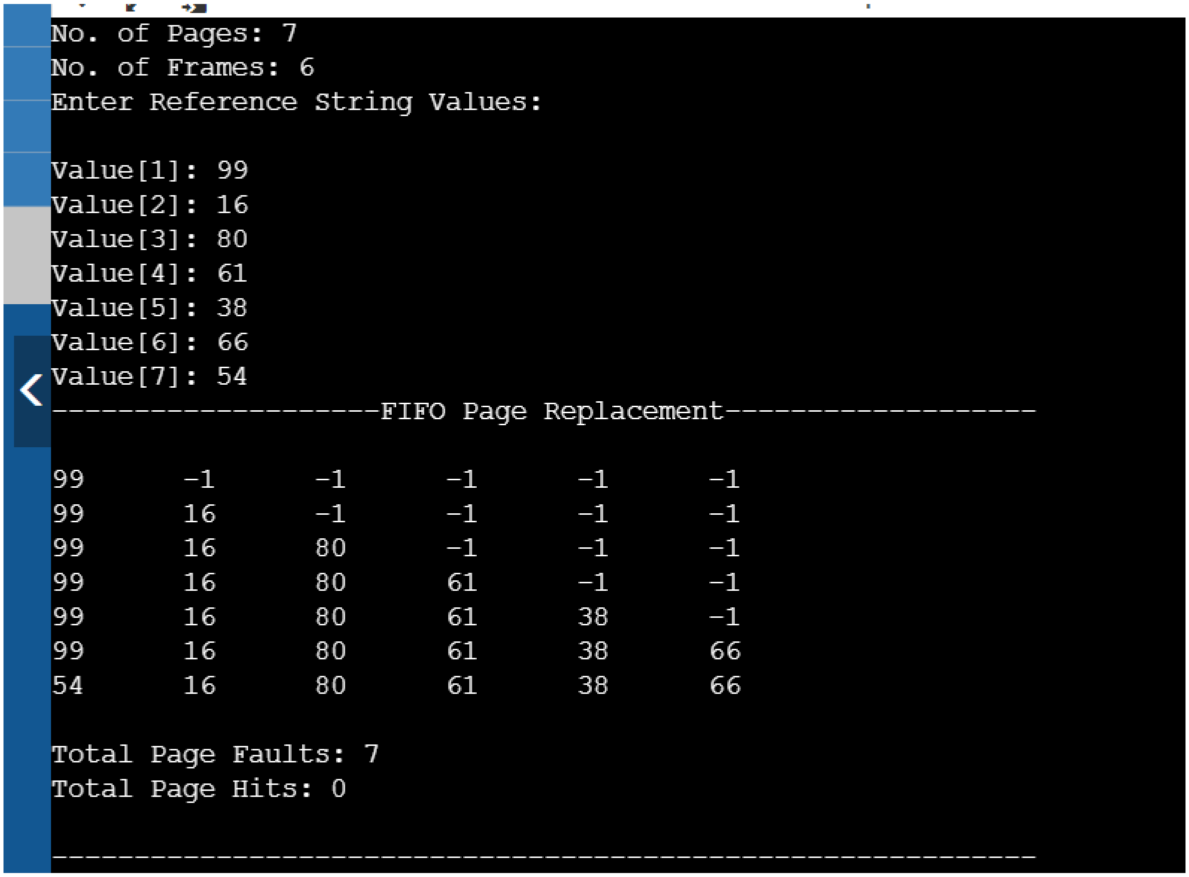


```
No. of Pages: 5No. of Frames: 8
Enter Reference String Values:
Value[1]: 19Value[2]: 46Value[3]: 84Value[4]: 46Value[5]: 11
--------------------FIFO Page Replacement-------------------

19      -1      -1      -1      -1      -1      -1      -1
19      46      -1      -1      -1      -1      -1      -1
19      46      84      -1      -1      -1      -1      -1
19      46      84      -1      -1      -1      -1      -1
19      46      84      -1      11      -1      -1      -1

Total Page Faults: 4
Total Page Hits: 1

------------------------------------------------------------
--------------------LRU Page Replacement--------------------

        19
19      46
19      46      84
19      46      84      11


Total Page Faults: 4
Total Page Hits: 1

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

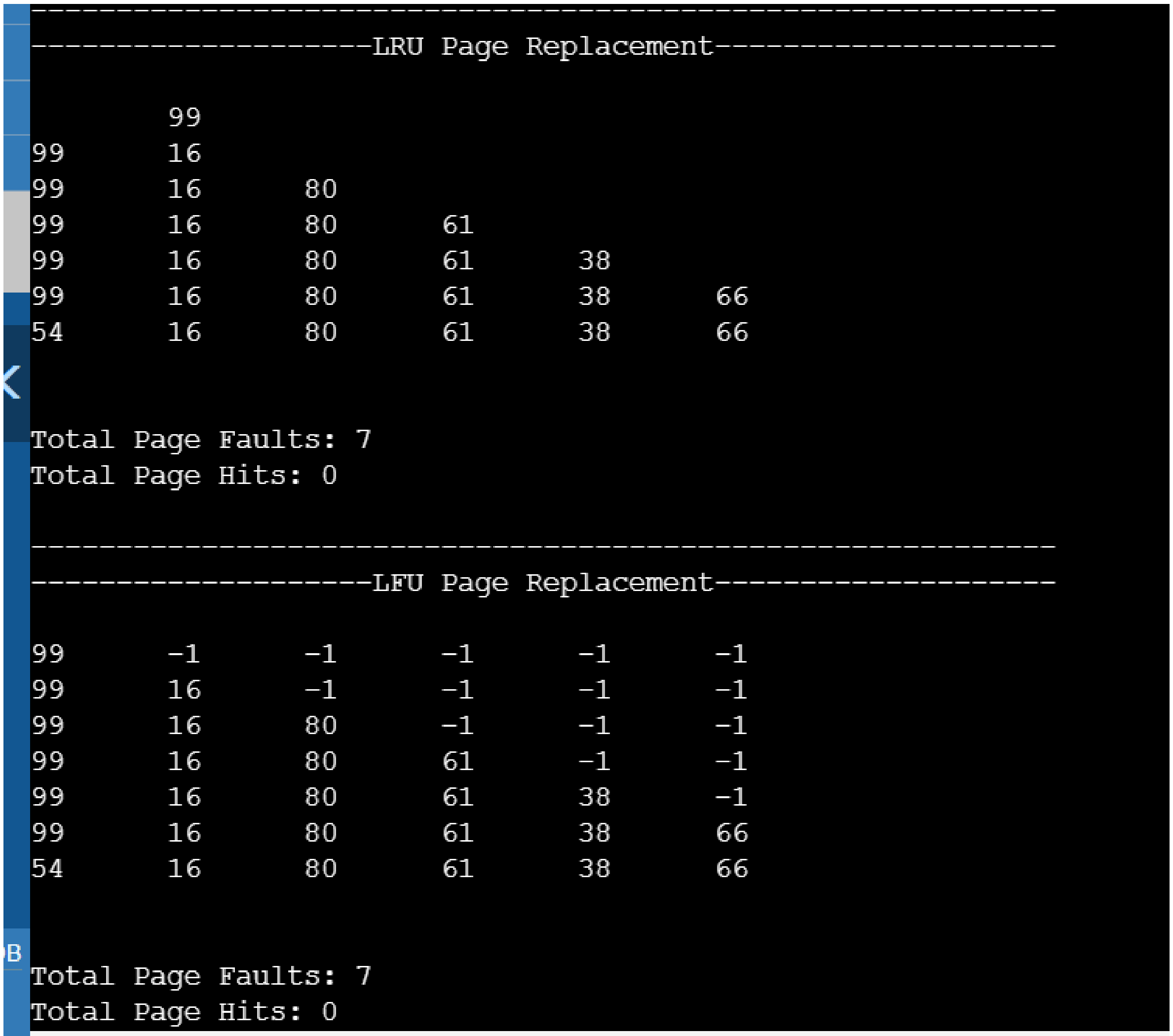


```
------------------------------------------------------------
--------------------LFU Page Replacement--------------------

19      -1      -1      -1      -1      -1      -1      -1
19      46      -1      -1      -1      -1      -1      -1
19      46      84      -1      -1      -1      -1      -1
19      46      84      -1      -1      -1      -1      -1
19      46      84      11      -1      -1      -1      -1


Total Page Faults: 4
Total Page Hits: 1

------------------------------------------------------------
------------------Optimal Page Replacement------------------

19      -1      -1      -1      -1      -1      -1      -1
19      46      -1      -1      -1      -1      -1      -1
19      46      84      -1      -1      -1      -1      -1
19      46      84      -1      -1      -1      -1      -1
19      46      84      11      -1      -1      -1      -1

Total Page Faults: 4
Total Page Hits: 1

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

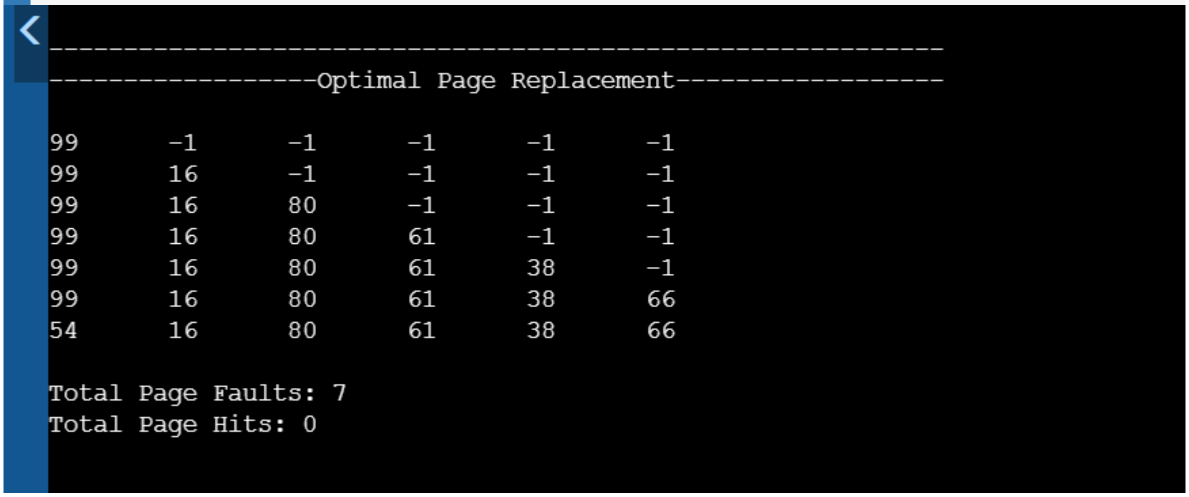
Second Time Run:



```
No. of Pages: 7
No. of Frames: 6
Enter Reference String Values:

Value[1]: 99
Value[2]: 16
Value[3]: 80
Value[4]: 61
Value[5]: 38
Value[6]: 66
Value[7]: 54
--------------------FIFO Page Replacement-------------------

99      -1      -1      -1      -1      -1
99      16      -1      -1      -1      -1
99      16      80      -1      -1      -1
99      16      80      61      -1      -1
99      16      80      61      38      -1
99      16      80      61      38      66
54      16      80      61      38      66

Total Page Faults: 7
Total Page Hits: 0

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



```
------------------------------------------------------------
--------------------LRU Page Replacement--------------------

        99
99      16
99      16      80
99      16      80      61
99      16      80      61      38
99      16      80      61      38      66
54      16      80      61      38      66


Total Page Faults: 7
Total Page Hits: 0

------------------------------------------------------------
--------------------LFU Page Replacement--------------------

99      -1      -1      -1      -1      -1
99      16      -1      -1      -1      -1
99      16      80      -1      -1      -1
99      16      80      61      -1      -1
99      16      80      61      38      -1
99      16      80      61      38      66
54      16      80      61      38      66


Total Page Faults: 7
Total Page Hits: 0
```



```
------------------------------------------------------------
------------------Optimal Page Replacement------------------

99      -1      -1      -1      -1      -1
99      16      -1      -1      -1      -1
99      16      80      -1      -1      -1
99      16      80      61      -1      -1
99      16      80      61      38      -1
99      16      80      61      38      66
54      16      80      61      38      66

Total Page Faults: 7
Total Page Hits: 0
```