```
// FIFO Page Replacement
#include <bits/stdc++.h>
using namespace std;
const int N=100005;
void fifo_page_replacement(int frame_size, int n,int pages[])
{
    int mark[N];     queue<int> Q;
    int page_faults=0;
    for(int i=0; i<n; i++)
    {
        if(mark[pages[i]]==true)
        {
        }         else
        {
            Q.push(pages[i]);
            mark[pages[i]]=true;
            if(Q.size()>frame_size)
            {
                int p= Q.front();
     mark[p]=false;
                Q.pop();
            }
            page_faults++;
        } }
 cout<<"Frame size  Page faults  Page Hits\n";
 cout<<"  "<<frame_size<<" "<<page_faults<<" "<<npage_faults<<"\n";
     return;
}
int main()
{
int frame_size=4; int pages[N];
int n;
cout<<"Page Reference Stream Length: ";     cin>>n;
cout<<"Page Reference Stream:\n";     for(int i=0; i<n; i++)
cin>>pages[i];
fifo_page_replacement(frame_size,n,pages);
return 0;
}
```

```
/tmp/ZOF0x6dE39.o
Page Reference Stream Length: 20
Page Reference Stream:
5 5 3 4 7 4 5 6 9 8 7 4 5 6 5 1 0 2 3 5
Frame size          Page faults          Page Hits
     4                    15                5
```

```
//LRU Page Replacement
#include<iostream>
using namespace std;
int lru(int time[], int n){ int i, min = time[0], pos = 0;
for(i = 1; i < n; ++i)
{
     if(time[i] < min)
{
     min  = time[i];
     pos = i;
} }
     return pos;
}
int main()
{
int frameno, pageno=20, frames[10], page[30], count = 0, time[10],
flag1, flag2, i, j, pos, pf = 0,hit=0;
cout<<"Enter number of frames: "; cin>>frameno; cout<<"Enter 20
pages: "; for(i = 0; i < pageno; ++i)
{
cin>>page[i];
}
     for(i = 0; i < frameno; ++i)
{
frames[i] = -1;
}
for(i = 0; i < pageno; ++i)
{
flag1 = flag2 = 0;
for(j = 0; j < frameno; ++j)
{
```

```
if(frames[j] == page[i])
{
count++; hit++; time[j] = count; flag1 = flag2 = 1; break;
} }
if(flag1 == 0)
{
for(j = 0; j < frameno; ++j)
{
if(frames[j] == -1)
{
count++; pf++; frames[j] = page[i]; time[j] = count; flag2 = 1;
break;
} } }
if(flag2 == 0)
{
pos = lru(time, frameno); count++; pf++; frames[pos] = page[i];
time[pos] = count;
} }
cout<<"\n\nNumber of Page Faults    "<<pf; cout<<"\nNumber of page
hits   "<<hit<<"\n";;
return 0;
}
```

```
/tmp/ZOF0x6dE39.o
Enter number of frames: 4
Enter 20 pages: 5 5 3 4 7 4 5 6 9 8 7 4 5 6 5 1 0 2 3 5
Number of Page Faults    16
Number of page hits   4
```

// Optimal Page Replacement

```
#include <bits/stdc++.h>
using namespace std;
bool search(int key, vector<int>& fr)
{
     for (int i = 0; i < fr.size(); i++)
          if (fr[i] == key)
               return true;
     return false;
}
int predict(int pg[], vector<int>& fr, int pn, int index)
{
     int res = -1, farthest = index;
```

```
        for (int i = 0; i < fr.size(); i++) {
            int j;
            for (j = index; j < pn; j++) {
                if (fr[i] == pg[j]) {
                    if (j > farthest) {
                        farthest = j;
                        res = i;
                    }
                    break;
                }
            }
            if (j == pn)
                return i;
        }
        return (res == -1) ? 0 : res;
}
void optimalPage(int pg[], int pn, int fn)
{
        vector<int> fr;
        int hit = 0;
        for (int i = 0; i < pn; i++) {
            if (search(pg[i], fr)) {
                hit++;
                continue;
            }
            if (fr.size() < fn)
                fr.push_back(pg[i]);
            else {
                int j = predict(pg, fr, pn, i + 1);
                fr[j] = pg[i];
        } }
        cout << "No. of hits = " << hit << endl;
        cout << "No. of faults = " << pn - hit << endl;
}
int main()
{
        int pg[] = { 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 };
        int pn = sizeof(pg) / sizeof(pg[0]);
        int fn = 4;
        optimalPage(pg, pn, fn);
        return 0;
}
```

```
/tmp/G39g1ACcjG.o
No. of hits = 7
No. of faults = 6
```

Edited By AP