Program 10

Problem Statement: Write a program to implement FIFO page replacement algorithm **(First In First Out).**

Overview: This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, 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.

C-Program:

```
#include<stdio.h>
int q[100];
int f=0,r=0;
int  check(int val)
{
   for(int i=0;i<r;i++)
   {
         if(q[i]==val)
              return 1;
   }
   return 0;
 }

int fault(int *pages,int n,int nf)
{
   int pf=0;
   int temp=f;
   for(int i=0;i<n;i++)
   {
   if(r<nf)
   {
      if(!check(pages[i]))
      {
            q[r]=pages[i];
            r++;
            pf++;

      }
    }
    else
    {
          if(!check(pages[i]))
         {
            q[temp++]=pages[i];
             pf++;
```

```
            for(int i=0;i<r;i++)
               printf("%d ",q[i]);
            printf("\n");
          }
      }
       for(int i=0;i<r;i++)
               printf("%d ",q[i]);
            printf("\n");
   }
   return pf;
}



int main()
{
   int n,nf;
   printf("Achintya Mishra \t Section: A \t 20011857 \n");
   printf("Enter the number of pages: ");
   scanf("%d",&n);
   int pages[n];
   printf("Enter the pages: ");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&pages[i]);
   }
     printf("Enter the number of frames: ");
   scanf("%d",&nf);
   q[n];
   printf("Total Number of Page Faults: %d\n",fault(pages,n,nf));
   return 0;
}
```

OUTPUT

```
student@LAB1PC46:~/Desktop/OS$ gcc fifo.c
student@LAB1PC46:~/Desktop/OS$ ./a.out
Achintya Mishra          Section: A      20011857
Enter the number of pages: 13
Enter the pages: 7 0 1 2 0 3 0 4 2 3 0 3 2
Enter the number of frames: 4
7
7 0
7 0 1
7 0 1 2
7 0 1 2
3 0 1 2
3 0 1 2
3 0 1 2
3 4 1 2
3 4 1 2
3 4 1 2
3 4 1 2
3 4 0 2
3 4 0 2
3 4 0 2
3 4 0 2
Total Number of Page Faults: 7
student@LAB1PC46:~/Desktop/OS$
```

Program 11

Problem Statement: Write a program to implement LRU Algorithm.

Overview: 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.

C-Program:

```
#include<stdio.h>
#include<limits.h>
int q[100];
int f=0,r=0;
int  check(int val)
{
   for(int i=0;i<r;i++)
   {
        if(q[i]==val)
            return 1;
   }
   return 0;
 }

int fault(int *pages,int n,int nf,int *ind)
{
   int pf=0;
   printf("\nQueue Changes As follows:\n");
   for(int i=0;i<n;i++)
   {
   if(r<nf)
   {
      if(!check(pages[i]))
      {
            q[r]=pages[i];
            r++;
            pf++;

      }
      ind[pages[i]]=i;
    }
    else
    {
         if(!check(pages[i]))
         {
```

```
            int t=INT_MAX,pos=-1;
            for(int i=0;i<nf;i++)
            {
                  if(ind[q[i]]<t)
                  {
                     t=ind[q[i]];
                     pos=i;
                   }
              }
              q[pos]=pages[i];
             pf++;
          }
          ind[pages[i]]=i;
      }
       for(int i=0;i<r;i++)
              printf("%d ",q[i]);
          printf("\n");
   }
   return pf;
}

int main()
{
   int n,nf;
   printf("Achintya Mishra \t Section: A \t 20011857 \n");
   printf("Enter the number of pages: ");
   scanf("%d",&n);
   int pages[n];
   printf("Enter the pages: ");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&pages[i]);
   }
    printf("Enter the number of frames: ");
   scanf("%d",&nf);
   q[nf];
   int ind[n];
   printf("\nTotal Number of Page Faults: %d\n",fault(pages,n,nf,ind));
   return 0;
}
```

OUTPUT

```
student@LAB1PC46:~/Desktop/OS$ gcc lru.c
student@LAB1PC46:~/Desktop/OS$ ./a.out
Achintya Mishra          Section: A      20011857
Enter the number of pages: 13
Enter the pages: 7 0 1 2 0 3 0 4 2 3 0 3 2
Enter the number of frames: 4

Queue Changes As follows:
7
7 0
7 0 1
7 0 1 2
7 0 1 2
3 0 1 2
3 0 1 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2

Total Number of Page Faults: 6
```