Lab-3:

Introduction to Page Replacement Algorithm:

Theory:

The page replacement algorithm decides which memory page is to be replaced.

Page Replacement FIFO:

It is the simplest page replacement algorithm. In this algorithm, the OS keeps tracks 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.

Page Replacement LRU:

In this algorithm, page will be replaced which is least recently used. Pages that have not been used for ages will be probably remain unused for long time. It is not cheap to implement LRU, it is necessary to maintain a linked list of all pages in memory with the Most Recently used page at the front and Least Recently used page at the rear.

Program 1

Source code: (fif.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int i,j,nof,nor, flag=0, ref[50], frm[50], pf=0, victim=-1;
void main()
{
    printf("\n\t\t\t FIFO PAGE REPLACEMENT ALGORITHM");
    printf("\n Enter no. of frames ....");
    scanf("%d",&nof);
    printf(" Enter number of Pages \n");
    scanf("%d", &nor);
    for(i=0; i<nor; i++)
    {
        printf("\n Enter the Page No...");
        scanf("%d", &ref[i]);
    }
    printf("\n The given Pages are:");
    for(i=0; i<nor; i++)
    {
        printf("%4d", ref[i]);
    }
    for(i=1; i<=nof; i++)
    {
        frm[i] = -1;
        printf("\n");
    }
    for(i=0; i<nor; i++)
    {
        flag=0;
        printf("\n\t page no %d ->\t", ref[i]);
        for(j=0; j<nof; j++)
        {
            if (frm[j]==ref[i])
            {
                flag=1;
                break;
            }
        }
        if (flag==0)
        {
            pf++;
            victim++;
```

```
            victim = victim %nof;
            frm[victim] = ref[i];
            for(j=0; j<nof; j++)
            {
                printf("%4d", frm[j]);
            }
        }
    }
    printf("\n\n\t\t No. of pages faults ... %d", pf);
}
```

Input and Output:

Input:

```
$cc -c fif.c -o fif.o
$cc fif.o -o fif
$./fif
```

Output:

```
        FIFO PAGE REPLACEMENT ALGORITHM
No. of frames ... 3
Enter no. of pages 12
Enter the page no ..5
Enter the page no ..0
Enter the page no ..5
Enter the page no ..3
Enter the page no ..5
Enter the page no ..2
Enter the page no ..5
Enter the page no ..0
Enter the page no ..1
Enter the page no ..0
Enter the page no ..7
Enter the page no ..3

The given pages are : 5 0 5 3 5 2 5 0 1 0 7 3
page no. 5 ->     5    -1    -1
page no. 0 ->     5     0    -1
page no. 5 ->
page no. 3 ->     5     0     3
page no. 5 ->
page no. 2 ->     2     0     3
page no. 5 ->     2     5     3
page no. 0 ->     2     5     0
page no. 1 ->     1     5     0
page no. 0 ->
page no. 7 ->     1     7     0
page no. 3 ->     1     7     2
```

No. of page faults --- 9

Discussion:

In this program, we used the first in first out page replacement algorithm. The replacement is done on the basis of queue of pages in memory.

Program 2:

Source code (lru.c)

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int i, j, nof, nor, flag=0, ref[50], frm[50], pf=0, victim=-1;
int recent[10], lrucal[50], count=0;
int lruvictim();
void main()
{
    printf("\n\t\t\t LRU PAGE REPLACEMENT ALGORITHM");
    printf("\n Enter no. of Frames....");
    scanf("%d", &nof);
    printf("Enter no. of reference string --");
    scanf("%d", &nor);
    printf("\n Enter reference string --");
    for(i=0; i<nor; i++)
    {
        scanf("%d", &ref[i]);
    }
    printf("\n\n\t\t LRU PAGE REPLACEMENT ALGORITHM");
    printf("\n\t The given reference string:");
    printf("\n ----------");
    for(i=0; i<nor; i++)
    {
        printf("%4d", ref[i]);
    }
    for(i=1; i<=nof; i++)
    {
        frm[i]=-1;
        lrucal[i]=0;
    }
    for(i=0; i<10; i++)
    {
        recent[i]=0;
        printf("\n");
    }
```

```
for(i=0;i<nor;i++)
{
    flag=0;
    printf("\n\t Reference No %d -> \t",ref[i]);
    for(j=0;j<nof;j++)
    {
        if(frm[j]==ref[i])
        {
            flag=1;
            break;
        }
    }
    if(flag==0)
    {
        count++;
        if(count<=nof)
            victim++;
        else
            victim=lruvictim();
            pf++;
            frm[victim]=ref[i];
            for(j=0;j<nof;j++)
            {
                printf("%4d",frm[j]);
            }
        }
        recent[ref[i]]=i;
    }
    printf("\n\n\t No. of page faults---%d",pf);
    getch();
}
int lruvictim()
{
    int i,j,temp1,temp2;
    for(i=0;i<nof;i++)
    {
        temp1=frm[i];
        lrucal[i]=recent[temp1];
    }
    temp2=lrucal[0];
    for(j=1;j<nof;j++)
    {
        if(temp2>lrucal[j])
        {
            temp2=lrucal[j];
```

```
    for(i=0; i<nof; i++)
    {
        if (ref[temp2]==frm[i])
            return i;
            return 0;
    }
}
```

Input and Output:

Input:

```
gcc -c lru.c -o lru.o
gcc lru.o -o lru
./lru
```

Output:

```
        LRU PAGE REPLACEMENT ALGORITHM
Enter no. of frames ....3
Enter no. of reference string ...12
Enter reference string ...5
0
5
3
5
2
5
0
1
0
7
3
```

```
        LRU PAGE REPLACEMENT ALGORITHM
The given reference string ----------
5  0  5  3  5  2  5  0  1  0  7  3

page no 5 -> 5  -1  -1
page no 0 -> 5  0   -1
page no 5 ->
page no 3 -> 5  0   3
page no 5 ->
page no 2 -> 2  0   3
```

```
page no 5 -> 5      0    3
page no 0 ->
page no 1 -> 1      0    3
page no 0 ->
page no 7 -> 7      0    3
page no 3 ->
```

No. of page faults ... 7

Discussion

In this program, we used least recently used page replacement algorithm. The page that is least recently used is replaced.

Conclusion

In this lab, we used FIFO and LRU page replacement algorithm.