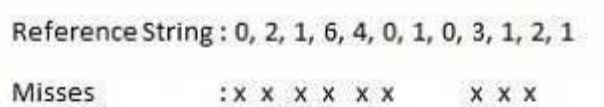
Name: Shawn
Louis
SE COMPS
Roll No: 31

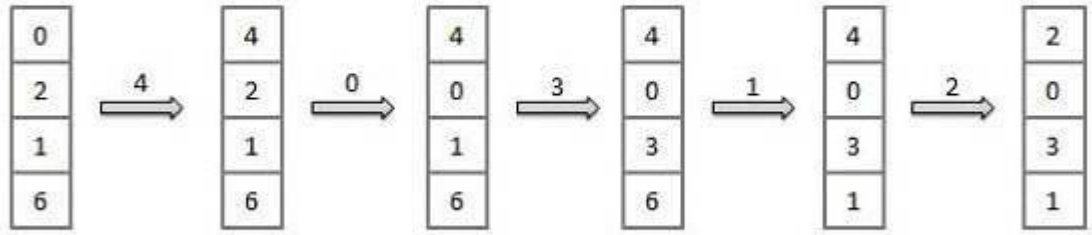
Experiment no 8

Aim : Write a program to implement various page replacement policies FIFO and LRU algorithms

Theory :
FIFO (First in First out) Page Replacement Algorithm − It is one of the simplest page replacement algorithm. The oldest page, which has spent the longest time in memory is chosen and replaced. This algorithm is implemented with the help of FIFO queue to hold the pages in memory. A page is inserted at the rear end of the queue and is replaced at the front of the queue.



In the fig., the



Fault Rate = 9 / 12 = 0.75

reference string is 5, 4, 3, 2, 5, 4, 6, 5, 4, 3, 2, 6 and there are 3 frames empty. The first 3 reference (5, 4, 3) cause page faults and are brought into empty frames. The next reference (2) replaces page 5 because page 5 was loaded first and so on. After 7 page faults, the next reference is 5 and 5 is already in memory so no page fault for this reference. Similarly for next reference 4. The + marks shows incoming of a page while circle shows the page chosen for removal.

Advantages

- FIFO is easy to understand.
- It is very easy to implement.

Disadvantage

- Not always good at performance. It may replace an active page to bring a new one and thusmay cause a page fault of that page immediately.
- Another unexpected side effect is the FIFO anomaly or Belady's anomaly. This anomaly says thatthe page fault rate may increase as the number of allocated page frames increases.

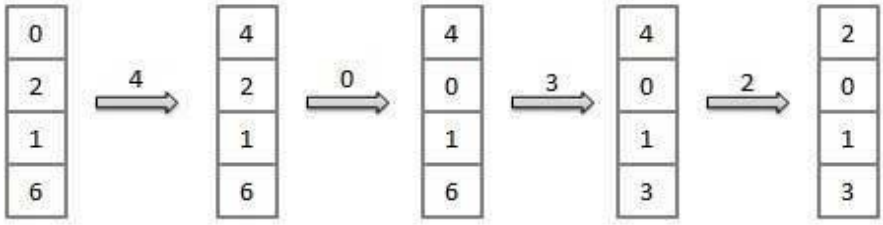
LRU :
LRU(Least Recently Used) Algorithm − The Least Recently used (LRU) algorithm replaces the page that has not been used for the longest period of time. It is based on the observation that pages that have notbeen

used for long time will probably remain unused for the longest time and are to be replaced.

Program :
FIFO

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Initially, 3 page frames are empty. The first 3 references (7, 0, 1) cause page faults and are brought into these empty frames. The next reference (2) replaces page 7. Since next page reference (0) is already in memory, there is no page fault. Now, for the next reference (3), LRU replacement sees that, of the three frames in memory, page 1 was used least recently, and thus is replaced. And thus the process continues.

Advantages

- LRU page replacement algorithm is quiet efficient.
- It does not suffer from Belady's Anomaly.

Disadvantages

- Its implementation is not very easy.
- Its implementation may require substantial hardware assistance.

Output :

Program 1 : FIFO

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
        printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
        printf("\n ENTER THE PAGE NUMBER :\n");
        for(i=1;i<=n;i++)
        scanf("%d",&a[i]);



   }
```

```
        printf("\n ENTER THE NUMBER OF FRAMES :");
        scanf("%d",&no);
for(i=0;i<no;i++)
        frame[i]= -1;
                j=0;
                printf("\tref string\t page frames\n");
for(i=1;i<=n;i++)
                {
                        printf("%d\t\t",a[i]);
                        avail=0;
                        for(k=0;k<no;k++)
if(frame[k]==a[i])
                                avail=1;
                        if (avail==0)
                        {
```

```
    }
```

```
                        frame[j]=a[i];
                        j=(j+1)%no;
                        count++;
                        for(k=0;k<no;k++)
                        printf("%d\t",frame[k]);
}
                printf("\n");
}
        printf("Page Fault Is %d",count);
        return 0;
}
```

```
ENTER THE NUMBER OF PAGES:  20
ENTER THE PAGE NUMBER :       7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
ENTER THE NUMBER OF FRAMES :3
      ref string       page frames
7               7       -1      -1
0               7       0       -1
1               7       0       1
2               2       0       1
0
3               2       3       1
0               2       3       0
4               4       3       0
2               4       2       0
3               4       2       3
0               0       2       3
3
2
1               0       1       3
2               0       1       2
0
1
7               7       1       2
0               7       0       2
1               7       0       1
Page Fault Is 15
}
```

Program 2 : LRU

```
#include<stdio.h>

int findLRU(int time[], int n){
   int i, minimum = time[0], pos = 0;

   for(i = 1; i < n; ++i){
      if(time[i] < minimum){
         minimum = time[i];
         pos = i;
      }

   }
```

```
    return pos;
}

int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults
= 0;
    printf("Enter number of frames: ");
    scanf("%d", &no_of_frames);

    printf("Enter number of pages: ");
    scanf("%d", &no_of_pages);

    printf("Enter reference string: ");

    for(i = 0; i < no_of_pages; ++i){
        scanf("%d", &pages[i]);
    }

    for(i = 0; i < no_of_frames; ++i){
        frames[i] = -1;
    }

    for(i = 0; i < no_of_pages; ++i){
        flag1 = flag2 = 0;

        for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == pages[i]){
                counter++;
                time[j] = counter;
                    flag1 = flag2 = 1;
                    break;
                }
        }

        if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
                if(frames[j] == -1){
                    counter++;
                    faults++;
                    frames[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    break;
                }
            }
        }

        if(flag2 == 0){
            pos = findLRU(time, no_of_frames);
            counter++;
            faults++;
            frames[pos] = pages[i];
            time[pos] = counter;
    }
```

```
        printf("\n");

        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }

    printf("\n\nTotal Page Faults = %d", faults);

    return 0;
}
```

```
Enter number of frames: 3
Enter number of pages: 6
Enter reference string: 5 7 5 6 7 3

5 -1 -1
5 7 -1
5 7 -1
5 7 6
5 7 6
3 7 6

Total Page Faults = 4
```

Conclusion :

Thus various page replacement policies

using FIFO and LRU algorithms were

implemented in C language and their

advantages and disadvantages were

pondered upon using various examples

and studying them diagrammatically as

well.