Ex. No.: 11a)
Date: 16.4.25

FIFO PAGE REPLACEMENT

Aim:

To find out the number of page faults that occur using First-in First-out (FIFO) page replacement technique.

Algorithm:

1. Declare the size with respect to page length
2. Check the need of replacement from the page to memory
3. Check the need of replacement from old page to new page in memory 4.
Form a queue to hold all pages
5. Insert the page require memory into the queue
6. Check for bad replacement and page fault
7. Get the number of processes to be inserted
8. Display the values

Program Code:

```
#include <stdio.h>
int main()
    int refstr[100], frame[10];
    int refstr, framesize;
    int index=0; isfit.pf=0;
    printf("Enter size of ref string:");
    scanf("%d", &refsize);
    for(int i=0; i<=refsize; i++){
            printf("Enter [%d]:", i++){
            scanf("%d", &refstr[i]);
}
printf("Enter page name size:");
scanf("%d", &name size:");
for(int i=0; i<refsize; i++){
        id=0;
```

```
for (int j=0; j< framesize ; j++){
        if (frames [j] == refstr[i] {
            isHit=1;
            break;
        }
}

if (! isHit) {
    frames[index] = refstr [i];
    index = (index+1) %. namesize ;
    p++;
    printf("%.d →", refstr [i]);
    for (int k=0 ; k < framesize; k++){
        if (names [k] ! = -1)
            print("%.d ", names [k]);
        }
    print (" \n");
}
else else {
    printf("%.d → Nopage faults \n", refstr[i]);
}
}
printf ("\n Total page faults : %.d \n", pf );
}
```

Sample Output:

```
[root@localhost student]# python fifo.py

Enter the size of reference string: 20
Enter [ 1] : 7
Enter [ 2] : 0
Enter [ 3] : 1
Enter [ 4] : 2
Enter [ 5] : 0
Enter [ 6] : 3
Enter [ 7] : 0
Enter [ 8] : 4
Enter [ 9] : 2
Enter [10] : 3
Enter [11] : 0
Enter [12] : 3
Enter [13] : 2
Enter [14] : 1
Enter [15] : 2
Enter [16] : 0
Enter [17] : 1
Enter [18] : 7
Enter [19] : 0
Enter [20] : 1

Enter page frame size : 3

7 -> 7 - -
0 -> 7 0 -
1 -> 7 0 1
2 -> 2 0 1
0 -> No Page Fault


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 -> No Page Fault
2 -> No Page Fault
1 -> 0 1 3
2 -> 0 1 2
0 -> No Page Fault
1 -> No Page Fault
7 -> 7 1 2
0 -> 7 0 2
```

1 ->7 0 1
Total page faults: 15.
[root@localhost student]#

OUTPUT :

Enter the size of ref string : 7

Enter page frame size : 3

Enter [1] : 1

Enter [2] : 3

Enter [3] : 0

Enter [4] : 3

Enter [5] : 5

Enter [6] : 6

Enter [7] : 3

1 → 1

3 → 13

0 → 130

3 → No page fault Total page Faults : 6

5 → 530 6 → 560 3 → 563

Result :

A program for finding the page fault using FIFO replacement.

Ex. No.: 11b)
Date: 18.4.25

LRU

Aim:

To write a c program to implement LRU page replacement algorithm.

Algorithm:

1: Start the process
2: Declare the size
3: Get the number of pages to be inserted
4: Get the value
5: Declare counter and stack
6: Select the least recently used page by counter value
7: Stack them according the selection.
8: Display the values
9: Stop the process

Program Code:

```
#include <stdio.h>
int main() {
    int refstr[100], frame[10]; recent[20];
    int refstr, framesize;
    int index=0; isHit pf=0,j,i,k,time=0;
    printf("Enter no: of pages");
    scanf("%d", &ref size);
    for (int i=0; i<=refsize; i++) {
        printf("Enter [%d]:", i++);
        scanf("%d", &refstr[i]);
    }
    printf("Enter page name size:");
    scanf("%d", &namesize);
    for (i=0; i<namesize; i++) {
        name[i] = -1;
        recent[i] = -1;
}
```

```
print ("\n");
for (int i=0; i< refsize ; i++){
        is Hit= 0;
      for (int j=0 ; j< namesize ; j++){
        if (name [j] == refstr [i]){
            isHit = 1;
            recent [j] = time++;
            break;
        } }
}
if (is Hit){
    printf ("%d →No: page fault\n", refstr
                                    [i]);
    continue;
}
int empty id = -1
for (j=0; j< frame size ; j++) {
        if [name [j]==-1] {
            empty id = j ;
            break;
        }
    }
    if (empty ind != -1){
        frames [empty ind] = refstr [i] ;
        recent [empty ind] = time ++;
} else {
int min = recent [0];
true index= 0;
```

```
for (j=1 ; j< framesize ; j++) {
    if (recent [j] < min) {
        min = recent [j];
        true index = j;
    }
}
frames [true index] = refstr [i];
recent [true index] = time++;
}
pf ++;
printf ("%d → ", ref str[i]);
for (int k=0 ; k < frame size ; k++){
    if (frames [k] != -1)
        printf ("%d" frames [k]);
}
printf ("→ page fault \n");
}
printf ("\n Total page faults: %d\n",
        pf);
}
```

Sample Output :

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

OUTPUT:

Enter no: of pages: 4
Enter [1] = 7
Enter [2] = 0
Enter [3] = 1
Enter [4] = 2
Enter [5] = 0
Enter [6] = 3
Enter [7] = 0
Enter [8] = 4
Enter [9] = 2
Enter [10] = 3
Enter [11] = 0
Enter [12] = 3
Enter [13] = 2
Enter [14] = 3

Enter page frame : 4
7 → 7 ⇒ page fault
0 → 70 ⇒ page fault
1 → 701 ⇒ page fault
2 → 712 ⇒ page fault
0 → No page fault
4 → 3042 ⇒ page fault
2 → No page fault
3 → No page fault
0 → No page fault
3 → No page Fault
2 → No page fault
3 → No page Fault

Total page Fault = 6

Result:

A C program for finding the page fault using page replacement technique is implemented successfully.

Ex. No.: 11c)
Date:

Optimal

Aim:
To write a c program to implement Optimal page replacement algorithm.

ALGORITHM:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least frequently used page by counter value
7. Stack them according the selection.
8. Display the values
9. Stop the process

PROGRAM:

```
# include <stdio.h>
int main(){
    unit f, p, fa[10], pa[30], temp[10], f1, f2, f3
                    i, j, k, pos, max, faults=0;
printf("Enter no: of frames");
Scanf("%d", &f);
printf("Enter the no:of pages");
Scanf("%d" ,&p);
printf("Enter the reprence string");
for(i=0; i<p ; i++){
        Scanf("%d", &pa[i]);
}
for(i=0 ; i <f ; i++){
        fa[i]= -1;
}
```

```
for (i=0 ; i<p; i++) {
    flag 1 = f2=0;
    for (j=0 ; j < f ;j++){
        if ( f a[j] == pa[j] ){
                f1 = f2=1;
        }
    }
break;
if (f1 ==0) {
    for (j=0 ; j < f ; j++) {
        if (fa[j] ==-1) {
            faults ++ ;
            fa [j] = pa[i];
            f2 = 1;
            break ;
        }
    }
}
if (f2==0) {
    f3=0;
    for (j=0 ; j <f ; j++) {
        temp[j] = -1;
        for (k = i+1; k<p ; k++)
        {
            if (fa [j] = p a[k])
            {
                temp[j]=k;
```

```
                break;
            }
        }
    }
    for (j=0 ; j <nf ;j++){
        if (temp[j] ==-1) {
            pod = j;
            f3 = 1;
            break;
        }
    }
    if (f3 == 0) {
        max = temp [0];
        pod = 0;
        for (j=1 ; j < f; j++){
            if (temp[j] > max){
                max= temp[j] {
                pod = j
            }
        }
    }
    frames [pod] = pa[i];
            faults++;
}
```

```
        printf ("\n");
        for (j=0; j<f; j++){
            printf ("%d\t", fa[j]);
        }
    }
    printf ("\n\n = %d", Faults);
    return;
}
```

Output:

3
10
2 3 4 2 1 3 7 5 4 3
2 - 1 - 1
2 - 3 - 1
2 - 3 - 4
2 - 3 - 4
1 - 3 - 4
1 - 3 - 4
1 - 3 - 4
7 - 3 - 4
5 - 3 - 4
5 - 3 - 4
5 - 3 - 4
page fault - 4

Result:

Thus the code for optimal page replacement algorithm is executed successfully.