Ex. No.: 11a)
Date: 12/4/25

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 f, n, index=0, pf=0;
    printf("Enter the size of reference string:");
    scanf("%d", &n);
    int r[n];
    for (int i=0; i<n; i++){
        printf("Enter [%d]", i+1);
        scanf("%d", &r[i]);
    }
    printf("Enter page frame size:");
    scanf("%d", &f);
    int fr[f];
    for (int i=0; i<f; i++)
        fr[i] = -1;
```

```
int found;
for (int i=0; i<n ; i++) {
    found = 0;
    printf ("%d -> " , r[i]);
    for (int j= 0; j<f ; j++) {
        if (fr[j] == r[i]) {
            found = 1;
            printf ("No Page Fault");
            break;
        }
    }
    if (!found) {
        fr[index] = r[i];
        index = (index + 1)%f;
        pf++;
        for (int k = 0 ; k < f ; k++) {
            if (fr[k] != -1)
                printf ("%d", fr[k]);
            else
                printf ("- ");
        }
    }
    printf ("\n");
}
printf ("Total Page Fault : %d", 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]#
```

Result :

Program for finding the page fault using FIFO replacement has been executed succesfully

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 → 1 3

0 → 1 3 0 1

No page fault

5 → 5 3 0

6 → 5 6 0

3 → 5 6 3

Total page faults : 6

Ex. No.: 11b)
Date: 12/4/25

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 findLRU(int t[],int n){
    int i, min=t[0], pos=0;
    for(i=1;i<n;++i){
        if(t[i]<min){
            min=t[i];
            pos=i;
        }
    }
    return pos;
}
int main(){
    int f,n,i,j,pos,pf=0,c=0,found;
    printf("Enter number of frames:");
    scanf("%d",&f);
    printf("Enter number of pages:");
    scanf("%d",&n);
```

```
int p[n], fr[f], t[f];
printf("Enter the page reference string:\n");
for (i=0; i<n; i++) {
    scanf("%d", &p[i]);
}
for (i=0; i<n; i++){
    fr[i]=-1;
}
for (i=0; i<n; i++){
    found = 0;
    for (j=0; j<f; j++){
        if (fr[j]==p[i]) {
            c++;
            t[j]= c;
            found = 1;
            break;
        }
    }
    if (!found) {
        for (j=0; j<f; j++){
            if (fr[j]==-1) {
                c++;
                pf++;
                fr[j]= p[i];
                t[j]= c;
                found = 1;
                breale;
            }
        }
    }
```

```
        if (!found){
            pos = findLRU(t,f);
            c++;
            pf++;
            fr[pos] = p[i];
            t[pos] = c;
        }
        for (j=0; j<f; j++){
            printf("%d", fr[j]);
        }
        printf("\n");
    }
    printf("\nTotal 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
```

Result:

Thus the C program to find page faults using LRU page replacement algorithm was executed successfully.

Output :

Enter no. of frames : 4
Enter number of pages : 14

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

7 → 7

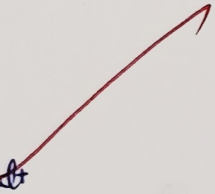
0 → 7 0

1 → 7 0 1

2 → 7 0 1 2

0 → No page fault

4 → 3 0 4 2



2 → No page fault

3 → No page fault

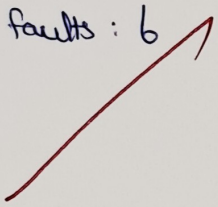
0 → No page fault

3 → No page fault

2 → No page fault

3 → No page fault

Total page faults : 6



Ex. No.: 11c)
Date: 18/4/25

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(){
    int n, f, i, j, k, pagefaults = 0, hit;
    printf("Enter the size of reference string:");
    scanf("%d", &n);
    int refstr[n];
    for (int i=0; i<n; i++){
        printf("Enter %d", i++);
        scanf("%d", &refstr[i]);
    }
    printf("Enter page frame size:");
```

```
scanf("%d", &f);
int frames[f];
for (i=0; i<f ; i++)
    frames[i] = -1;
for (i=0 ; i<n ; i++){
    hit =0;
    for (j=0 ; j<f ;j++) {
        if (frames[j] == refstr[i]){
            hit = 1;
            break;
        }
    }
    if (hit){
        printf("%2d -> No page fault \n", refstr[i]);
        continue;
    }
    int empty = -1;
    for (j=0 ; j<f ; j++){
        if (frames[j] == -1){
            empty=j;
            break;
        }
    }
```

```
if (empty != -1) {
    frames[empty] = refstr[i];
}
else {
    int farthest = -1, index = -1;
    for (j=0; j<f; j++) {
        int found = 0;
        for (k=i+1; k<n; k++) {
            if (frame[j] == refstr[k]) {
                found = 1;
                if (k > farthest) {
                    farthest = k;
                    index = j;
                }
                break;
            }
        }
        frames[index] = refstr[i];
    }
    pagefaults++;
    printf("%2d->", refstr[i]);
    for (k=0; k<f; k++) {
        if (frames[k] != -1)
            printf("%d", frames[k]);
        printf("-> pagefault\n");
    }
    printf("\nTotal page faults %d\n", pagefaults);
    printf("Total page hits: %d\n", n - pagefaults);
}
```

Result:

Thus the c program for optimal page replacement algorithm is executed successfully.

Output:

Enter the size of reference string: 10

Enter the page frame size: 3

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

7 → 7 => page fault

0 → 7 0 => page fault

1 → 7 0 1 => page fault

2 → 2 0 1 => page fault

0 → No page fault

3 → 2 0 3 => page fault

4 → 2 4 3 => page fault

2 → No page fault

3 → No page fault

Total page faults : 6

Total page hit : 4