Ex. No.: 11a)
Date: 18-4-25

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]#

Enter the size of reference 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

3 → No page fault

5 → 5 3 0

6 → 5 6 0

3 → 5 6 3

Total page fault : 6

Result :

A C program for ~~fin~~ FIFO page replacement is execute Sucessfully.

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 no of frames");
    scanf("%d", &f);
```

```
printf("Enter no 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<f; 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;
                break;
            }
        }
    }
}
```

```
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("\n Total page faults = %d\n", pf);
}
```

Enter no of frames : 3
Enter the page no 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 → 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 fault : 6.

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 for LRU page replacement is executed successfully.

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 ~~ref~~[100];
    int n, f, i, j, k, pf=0, hit;
    printf("Enter the size of reference string:");
    scanf("%d",&n);
    int rf[n];
    for(i=0;i<n;i++){
        printf("Enter [%d]:",i+1);
        scanf("%d",&rf[i]);
    }
    printf("Enter page frame size:");
    scanf("%d",&f);
    fr[f];
```

```
for(i=0; i<f; i++)
{
    fr[i] = -1;
}
for(i=0; i<n; i++){
    hit = 0;
    for(j=0; j<f; j++){
        if(fr[j] == rf[i]){
            hit = 1;
            break;
        }
    }
    if(hit){
        printf("%2d -> no page fault\n", rf[i]);
        continue;
    }
    int emp = -1;
    for(j=0; j<f; j++){
        if(fr[j] == -1){
            emp = j;
            break;
        }
    }
    if(emp != -1){
        fr[emp] = rf[i];
    }
    else{
        int far = -1; ind = -1
        for(j=0; j<f; j++){
            int found = 0;
            for(int k = i+1; k<n; k++){
```

```
            if (fr[j] == rf[k]) {
                found = 1
                if (k > far) {
                    far = k;
                    ind = j;
                }
                break;
            }
        }
        frames[ind] = rf[i];
    }
    pf++;
    printf("%2d →", rf[i]);
    for(k=0; k<f; k++){
        if (fr[k] != -1){
            printf("%d", fr[k]);
        }
        printf(" => pagefault\n");
    }
    printf("\n Total page faults = %d\n", pf);
    printf("Total page Hits: %d\n", n - pf);
}
```

Output:

```
Enter the size of reference string : 10
Enter the page frame : 3
Enter [1]:
Enter [2]:
Enter [3]:
Enter [4]:
Enter [5]:
Enter [6]:
Enter [7]:
Enter [8]:
Enter [9]:
Enter [10]:
```

7 → 7 ~~⇒ page~~
0 → 7 0
1 → 7 0 1
2 → 2 0 1
0 → NO page fault
3 → 2 0 3
4 → 2 4 3
2 → NO page fault
3 → NO page fault.

Total page fault : 6.
Total page hit : 4.

Result:

Thus the c program for optimal algorithm is executed successfully.