Practical-7

Q. Write a program to simulate Page replacement algorithm.

```
//  Program to simulate Page replacement algorithm: FIFO LRU and Optimal:-


// 1.FIFO And LRU:-


import java.util.*; public class Main {
public static void main(String[] args) {
 pr();
 }
 static void pr(){
 Scanner sc = new Scanner(System.in);
 System.out.println("Page Replacement : ");
 System.out.println("Enter 1 for FIFO ");
 System.out.println("Enter 2 for LRU ");
System.out.printf("Enter Choice : ");  int
x = sc.nextInt();
 System.out.printf("Length of String : " );
int n = sc.nextInt();


 int fr = 3;  int ref[] =
new int[n];  for (int i =
0; i < n; i++){ ref[i] =
sc.nextInt();
}
// FIFO
HashMap<Integer,Integer> map = new HashMap<>();
ArrayList<ArrayList<Integer>> arr = new ArrayList<>();
```

```
for(int i = 0 ; i <= n ; i++){
arr.add(new ArrayList<>());
}

for(int i = 0 ; i < fr ; i++){
 arr.get(0).add(-1);


}
 int ct = 1;  int hit = 0;  if(x ==
1 && n > 0){  int indx= 0;
for(int i = 1 ; i <= n ; i++){  int
curr = ref[i-1];
arr.get(i).addAll(arr.get(i-1));
if(!map.containsKey(curr)){
 if(indx < fr) arr.get(i).set((indx),ref[i-1]);
else{
 int min = Integer.MAX_VALUE;
 int temp = 0;  for(int j :
map.keySet()){
if(map.get(j) < min){
min = map.get(j);  temp
= j;
} }

for(int j = 0 ; j < fr ; j++){
if(arr.get(i).get(j) == temp){ arr.get(i).set(j,curr);
break;
}
}
```

```
map.remove(temp);
 }
 map.put(ref[i-1],ct++);
indx++;  }else{  hit++;
 }
 }
 }else if(x == 2 && n > 0 ){


 //LRU
 int indx= 0;  for(int i = 1 ; i <=
n ; i++){  int curr = ref[i-1];
arr.get(i).addAll(arr.get(i-1));
if(!map.containsKey(curr)){
 if(indx < fr) arr.get(i).set(indx,ref[i-1]);
else{
 int min = Integer.MAX_VALUE;
 int temp = 0; for(int j :
 map.keySet()){
 if(map.get(j) < min){
 min = map.get(j); temp
 = j;
}
}
for(int j = 0 ; j < fr ; j++){
if(arr.get(i).get(j) == temp){ arr.get(i).set(j,curr);
break;
 }
 }
 map.remove(temp);
 }
 indx++;
}else{  hit++;
 }
 map.put(ref[i-1],ct++);
 }
 }
System.out.println();
 for(int i = 0 ; i <= n ; i++){
for(int j = 0 ; j < fr ; j++){
 System.out.printf(arr.get(i).get(j) + " ");
 }
 System.out.println();
 }
System.out.println("Total Page Fault : " + (n - hit)); System.out.println("Total
Page Hit : " + hit);


sc.close();
}
}
```

#Ouput:

```
Page Replacement :
Enter 1 for FIFO
Enter 2 for LRU  Enter
Choice : 1
Length of String : 5
1
2
3
```

```
4
5

-1 -1                                    Error! Bookmark not defined.
1 -1                                     Error! Bookmark not defined.
1 2                                      Error! Bookmark not defined.
1 2                                      Error! Bookmark not defined.
4 2                                      Error! Bookmark not defined.
4 5                                      Error! Bookmark not defined.

Total Page Fault : 5
Total Page Hit : 0
```

```
// 2. Optimal Page Replacement Algorithm:-

import java.io.BufferedReader;
import java.io.IOException; import
java.io.InputStreamReader; public
class Main { public static void
main(String[] args) throws
IOException
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int rl, fr, pt = 0, hit = 0, fault = 0;
boolean isFull = false;
int buffer[]; int
```

```
reference[]; int
mem_layout[][];
System.out.println("\nENTER THE NUMBER OF FRAMES: ");
fr = Integer.parseInt(br.readLine());
System.out.println("\nENTER THE LENGTH OF REFERENCE STRING: ");
rl = Integer.parseInt(br.readLine());
reference = new int[rl];
mem_layout = new int[rl][fr]; buffer
= new int[fr];
for(int j = 0; j < fr; j++) buffer[j]
= -1;
System.out.println("\nENTER THE REFERENCE STRING: ");
for(int i = 0; i < rl; i++)
{
reference[i] = Integer.parseInt(br.readLine());
}
System.out.println();
for(int i = 0; i < rl; i++)
{
int search = -1;
for(int j = 0; j < fr; j++)
{
if(buffer[j] == reference[i])
{
search = j;
hit++; break;
}
}
if(search == -1)
```

```
{
if(isFull)
{
int index[] = new int[fr]; boolean
index_flag[] = new boolean[fr];
for(int j = i + 1; j < rl; j++)
{
for(int k = 0; k < fr; k++)
{
if((reference[j] == buffer[k]) && (index_flag[k] == false))
{
index[k] = j; index_flag[k]
= true; break;
}
}
}
int max = index[0];
pt = 0; if(max
== 0) max =
200; for(int j
= 0; j < fr;
j++)

{
if(index[j] == 0) index[j]
= 200; if(index[j] >
max)
{
max = index[j];
pt = j; }
}
}
buffer[pt] = reference[i]; fault++;
if(!isFull)
{
pt++;
if(pt == fr)
{
pt = 0; isFull
= true;
}
} }
for(int j = 0; j < fr; j++)
mem_layout[i][j] = buffer[j];
}
for(int i = 0; i < fr; i++) {
for(int j = 0; j < rl; j++) System.out.printf("%3d ",mem_layout[j][i]);
System.out.println();
}
System.out.println("\nTOTAL NUMBER OF HIT: " + hit);
System.out.println("\nHIT RATIO: " + (float)((float)hit/rl));
System.out.println("\nTOTAL NUMBER OF PAGE FAULT: " + fault);
}
}
```

#Output:

```
ENTER THE NUMBER OF FRAMES:
4
```

```
ENTER THE LENGTH OF REFERENCE STRING:

5


ENTER THE REFERENCE STRING:

1

2

3

4

5


 1   1   1   1   5

-1   2   2   2   2

-1  -1   3   3   3

-1  -1  -1   4   4

TOTAL NUMBER OF HIT: 0


HIT RATIO: 0.0


TOTAL NUMBER OF PAGE FAULT: 5
```