NAME:-Mayuresh Karnavat
ROLLNO:-13213

ASSIGNMENT NO.7

(Page Replacement Algorithms)

PROGRAM:-

```
//Page Replacement Algorithms
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;
import java.util.ArrayList;
import java.util.Scanner;
//Java implementation of FIFO page replacement
class FIFO
{ // Method to find page faults using FIFO
  public int pageFaults(int pages[], int n, int capacity)
  {     // To represent set of current pages. We use
    // an unordered_set so that we quickly check
    // if a page is present in set or not
    HashSet<Integer> s = new HashSet<>(capacity);
    // To store the pages in FIFO manner
    Queue<Integer> indexes = new LinkedList<>() ;
    // Start from initial page
    int page_faults = 0;
    for (int i=0; i<n; i++)
    {       // Check if the set can hold more pages
      if (s.size() < capacity)
      {           // Insert it into set if not present
        // already which represents page fault
        if (!s.contains(pages[i]))
        {s.add(pages[i]);
          // increment page fault
page_faults++;
          // Push the current page into the queue
indexes.add(pages[i]);          }       }       // If the set is full then need to perform FIFO
       // i.e. remove the first page of the queue from
       // set and queue both and insert the current page
       else
       {           // Check if current page is not already
        // present in the set
        if (!s.contains(pages[i]))
        {              //Pop the first page from the queue
          int val = indexes.peek();
indexes.poll();
          // Remove the indexes page
s.remove(val);
          // insert the current page
s.add(pages[i]);
```

NAME:-Mayuresh Karnavat
ROLLNO:-13213

```
            // push the current page into
            // the queue
indexes.add(pages[i]);
            // Increment page faults
page_faults++;          }       }     }
    return page_faults;  }
}//Java program for LRU page replacement algorithm
class LRU
{   // Driver method
  public void LRU_algo(int arr[], int capacity) {
     // To represent set of current pages.We use an Arraylist
ArrayList<Integer> s=new ArrayList<>(capacity);
     int count=0;
     int page_faults=0;
for(int i:arr)
     {         // Insert it into set if not present
        // already which represents page fault
        if(!s.contains(i))
        {         // Check if the set can hold equal pages
        if(s.size()==capacity)         {
s.remove(0);
s.add(capacity-1,i);         }
        else
s.add(count,i);
           // Increment page faults
page_faults++;
           ++count;         }
        else
        {            // Remove the indexes page
s.remove((Object)i);
           // insert the current page
s.add(s.size(),i);
        }      }System.out.println(page_faults);   }
}//Java program for Optimal page replacement algorithm
class Optimal {
  public void optimal_algo() {
     Scanner in = new Scanner(System.in);
     int frames = 0;
     int pointer = 0;
     int numFault = 0;
     int numhit = 0;
     int ref_len;
     boolean isFull = false; // Corrected variable declaration
     int buffer[];
     boolean hit[];
     int fault[];
     int reference[];
     int mem_layout[][];
```

NAME:-Mayuresh Karnavat
ROLLNO:-13213

```
System.out.println("Please enter the number of frames: ");
frames = Integer.parseInt(in.nextLine());
System.out.println("Please enter the length of the reference string: ");
ref_len = Integer.parseInt(in.nextLine());
reference = new int[ref_len];
mem_layout = new int[ref_len][frames];
buffer = new int[frames];
hit = new boolean[ref_len];
fault = new int[ref_len];
for (int j = 0; j < frames; j++) {
   buffer[j] = -1;
}      System.out.println("Please enter the reference string: ");
for (int i = 0; i < ref_len; i++) {
   reference[i] = in.nextInt();
}      System.out.println();
for (int i = 0; i < ref_len; i++) {
   int search = -1;
   for (int j = 0; j < frames; j++) {
      if (buffer[j] == reference[i]) {
         search = j;
         hit[i] = true;
         fault[i] = numFault;
         break;            }         }
   if (search == -1) {
      if (isFull) {
         int index[] = new int[frames];
         boolean index_flag[] = new boolean[frames]; // Corrected variable declaration
         for (int j = i + 1; j < ref_len; j++) {
            for (int k = 0; k < frames; k++) {
               if ((reference[j] == buffer[k]) && (index_flag[k] == false)) {
                  index[k] = j;
                  index_flag[k] = true;
                  break;                  }                  }               }
         int max = index[0];
         pointer = 0;
         if (max == 0) {
            max = 200;               }
         for (int j = 0; j < frames; j++) {
            if (index[j] == 0) {
               index[j] = 200;                  }
            if (index[j] > max) {
               max = index[j];
               pointer = j;                  }               }            }
      buffer[pointer] = reference[i];
      numFault++;
      fault[i] = numFault;
      if (!isFull) {
         pointer++;
```

NAME:-Mayuresh Karnavat
ROLLNO:-13213

```
            if (pointer == frames) {
               pointer = 0;
               isFull = true;              }          }        }
        for (int j = 0; j < frames; j++) {
           mem_layout[i][j] = buffer[j];        }     }
     for (int i = 0; i < ref_len; i++) {
        System.out.print(reference[i] + ": Memory is: ");
        for (int j = 0; j < frames; j++) {
           if (mem_layout[i][j] == -1) {
              System.out.printf("%3s ", "-1");
           } else {
              System.out.printf("%3d ", mem_layout[i][j]);          }        }
        System.out.print(": ");
        if (hit[i]) {
           System.out.print("Hit");
           numhit++;
        } else {          System.out.print("Page Fault");
           System.out.print(": (Number of Page Faults: " + fault[i] + ")");        }
        System.out.println();     }
     System.out.println("Total Number of Page Faults: " + numFault);
     System.out.println("Total Number of Hits: " + numhit);  }}
public class Page_replacement {
  public static void main(String args[]){
     int capacity;
     int n;
     FIFO fifo = new FIFO();
     LRU lru = new LRU();
     Optimal optimal = new Optimal();
     Scanner in = new Scanner(System.in);
     while(true){
System.out.println();
System.out.println("Menu");
System.out.println("1. FIFO ");
System.out.println("2. LRU");
System.out.println("3. Optimal");
System.out.println("4. exit");
System.out.println("Select the algorithm you want to implement: ");
        int choice = in.nextInt();
        switch(choice){
           case 1:
System.out.println("Enter the number of pages: ");
              n = in.nextInt();
              int pages[] = new int[n];
System.out.println("Enter the pages: ");
              for (int i = 0; i< n; i++){
                 pages[i] = in.nextInt();              }
System.out.println("Enter the capacity: ");
              capacity = in.nextInt();
```

```
System.out.println("FIFO Output");
            int faults = fifo.pageFaults(pages, n, capacity);
System.out.println("The number of page faults are: "+ faults);
            break;
          case 2:
System.out.println("Enter the number of pages: ");
            n = in.nextInt();
            int pages_lru[] = new int[n];
System.out.println("Enter the pages: ");
            for (int i = 0; i< n; i++){
pages_lru[i] = in.nextInt();            }
System.out.println("Enter the capacity: ");
            capacity = in.nextInt();
System.out.println("LRU Output");
lru.LRU_algo(pages_lru, capacity);
            break;
          case 3:
System.out.println("Optimal Output");
optimal.optimal_algo();
            break;
          case 4:
System.out.println("Exiting the code...");
            return;
          default:
System.out.println("Invalid option");      }    } }}
```

OUTPUT:-

```
Menu
1. FIFO
2. LRU
3. Optimal
4. exit
Select the algorithm you want to implement:
1
Enter the number of pages:
20
Enter the pages:
1
2
3
4
2
1
5
6
2
1
2
3
7
6
3
2
1
```

2
1
3
Enter the frames:
3
FIFO Output
The number of page hits are: 5
The number of page faults are: 15

Menu
1. FIFO
2. LRU
3. Optimal
4. exit
Select the algorithm you want to implement:
2
Enter the number of pages:
20
Enter the pages:
1
2
3
4
2
1
5
6
2
1
2
3
7
6
3
2
1
2
3
6
Enter the capacity:
3
LRU Output
15

Menu
1. FIFO
2. LRU
3. Optimal
4. exit
Select the algorithm you want to implement:
3
Optimal Output
Please enter the number of frames:
3
Please enter the length of the reference string:
20
Please enter the reference string:
1
2
3

NAME:-Mayuresh Karnavat
ROLLNO:-13213

4
2
1
5
6
2
1
2
3
7
6
3
2
1
2
3
6

1: Memory is: 1 -1 -1 : Page Fault: (Number of Page Faults: 1)
2: Memory is: 1 2 -1 : Page Fault: (Number of Page Faults: 2)
3: Memory is: 1 2 3 : Page Fault: (Number of Page Faults: 3)
4: Memory is: 1 2 4 : Page Fault: (Number of Page Faults: 4)
2: Memory is: 1 2 4 : Hit
1: Memory is: 1 2 4 : Hit
5: Memory is: 1 2 5 : Page Fault: (Number of Page Faults: 5)
6: Memory is: 1 2 6 : Page Fault: (Number of Page Faults: 6)
2: Memory is: 1 2 6 : Hit
1: Memory is: 1 2 6 : Hit
2: Memory is: 1 2 6 : Hit
3: Memory is: 3 2 6 : Page Fault: (Number of Page Faults: 7)
7: Memory is: 3 7 6 : Page Fault: (Number of Page Faults: 8)
6: Memory is: 3 7 6 : Hit
3: Memory is: 3 7 6 : Hit
2: Memory is: 3 2 6 : Page Fault: (Number of Page Faults: 9)
1: Memory is: 3 2 1 : Page Fault: (Number of Page Faults: 10)
2: Memory is: 3 2 1 : Hit
3: Memory is: 3 2 1 : Hit
6: Memory is: 6 2 1 : Page Fault: (Number of Page Faults: 11)
Total Number of Page Faults: 11
Total Number of Hits: 9

Menu
1. FIFO
2. LRU
3. Optimal
4. exit
Select the algorithm you want to implement:
4
Exiting the code...

NAME:-Mayuresh Karnavat
ROLLNO:-13213