```
//CODE FOR LDF PAGE REPLACEMENT ALGORITHM

#include<iostream>
#include <cstdlib>
#include <bits/stdc++.h>
using namespace std;

int max(int array[], int n){
   //int n = sizeof(array) / sizeof(array[0]);
   int m = array[0];
   for(int i = 1 ; i<n ; i++){
       if (array[i]>m){
           m = array[i];
       }
   }
   return m;
}
int min (int a, int b){
   if (a<b)
       return a;
   else
       return b;
}

int maxn (int a, int b){
   if (a>b)
       return a;
   else
       return b;
}
int distinct(int array[], int n , int dist[]){
   int m = max(array, n);
   int count[m+1]={0};
   for(int i = 0 ; i < n ; i++){
       count[array[i]]++;
   }
   int d = 0;
```

```
   for(int i = 0 ; i < m+1 ; i++){
       if(count[i]>0)d++;
   }
   //int dist[d];
   int c = 0;
   for(int i = 0 ; i < m+1 ; i++){
       if(count[i]>0){
           dist[c]=i;
           c++;
       }
   }
   return d;
}

int search(int temp[], int n, int p){
   int pos=-1;
   for(int i = 0 ;i <n ;i++){
       if (temp[i]==p)
           pos = i;
   }
   return pos;
}
int binarySearch(int arr[], int l, int r, int x)
{
   if (r >= l) {
       int mid = l + (r - l) / 2;
        if (arr[mid] == x)
           return mid;
       if (arr[mid] > x)
           return binarySearch(arr, l, mid - 1, x);

       return binarySearch(arr, mid + 1, r, x);
   }

   return -1;
}
int func(int temp[], int dist[], int x, int d, int n){
```

```
int d1 = binarySearch(dist,0,d-1,x);
int flag ;//1-clockwise
int m = -1;
int mar = -1;
int z;
int m1,m2;
for(int i = 0 ; i < n ; i++){
    if (temp[i] == x){return -1;}
    int d2 = binarySearch(dist,0,d-1,temp[i]);
    if(d2>d1){
        m1 = d2-d1;
        m2 = d2-d1-d;
    }
    else{
        m1 = d2-d1+d;
        m2 = d2-d1;
    }
    if (abs(m1)<abs(m2)){
        z = abs(m1);
        flag =1;
    }
    else{
        z= abs(m2);
        flag = 0;
    }


    //int z = min(abs(m1),abs(m2));
    if(z>=m){
        if(z==m){
            if(flag==0){
               m=z;
               mar = temp[i];
            }
        }
        else{
            m=z;
```

```
                mar = temp[i];
            }
        }

    }
    int pos = search(temp,n,mar);
    return(pos);
}
int countDistinct(int arr[], int n, int n5)
{

    unordered_set<int> s;
    int p;

    int res = 0;
    for (int i = 0; i < n; i++) {

        if (s.find(arr[i]) == s.end()) {
            s.insert(arr[i]);
            res++;
        }
        if(res==n5){
            p=i;
            break;
        }
    }
     return p;
}
int main(){
    cout<<"This is LDF page replacement algorithm"<<endl;

    int frames;
    cout<<"Enter the number of frames: ";
    cin>>frames;
    int n;
    cout<<"Enter the number of inputs: ";
    cin>>n;
```

```
cout<<"Enter the reference pages serially"<<endl;
int pages[n];
for(int i=0 ; i<n ; i++){
    cin>>pages[i];
}

int dist[n];
int d = distinct(pages,n,dist);
int c5 = countDistinct(pages,n,frames);
//cout<<c5<<endl;
cout<<endl;
int faults = 0;
int matrix[frames][n];
matrix[0][0]=pages[0];
for(int i = 1 ; i<frames;i++)
    matrix[i][0]=-1;

int si =0;
for (int j = 1 ; j < n ; j++){
    int temp[frames];
    int x = pages[j];

    if(pages[j]!=pages[j-1] &&si<frames){
        si++;
    }
    for(int i = 0 ; i < frames ; i++){

        if(j<=c5){
            //matrix[i][j]=matrix[i][j-1];

            if(i==si){
                matrix[i][j]=pages[j];
            }
            else
                matrix[i][j]=matrix[i][j-1];

        }
```

```
            //matrix[si][j]=pages[j];
            else{
                temp[i] = matrix[i][j-1];

                matrix[i][j] = matrix[i][j-1] ;
            }
        }
        if(j>c5){
            int pos1 = func(temp,dist,x,d,frames);
            if(pos1==-1){faults++;}
            matrix[pos1][j] = x;
        }

    }
    for(int i = 0 ; i < frames ; i++){
        for (int j = 0 ; j < n ; j++){
            if (matrix[i][j] == -1)
                cout<<"  ";
            else
                cout<<matrix[i][j]<<" ";
        }
        cout<<"\n";
    }
    cout<<"Page faults: "<<faults+frames+1;
}
```

INPUT/OUTPUT

This is LDF page replacement algorithm
Enter the number of frames: 3
Enter the number of inputs: 12
Enter the reference pages serially
0 1 2 3 0 1 4 0 1 2 3 4

0 0 0 0 0 0 0 0 0 2 2 2
 1 1 3 3 1 1 1 1 1 3 3
  2 2 2 2 4 4 4 4 4 4
Page faults: 7