**Name : Meenal Bharsakle**

**Class : SY-CS-C**

**Roll no : 20**

**LIFO**

#include <iostream>

using namespace std;

bool findA(int arr[],int a,int frame\_size){

    for(int i =0;i<frame\_size;i++){

        if(arr[i]==a){

            return true;

        }

    }

    return false;

}

void FIFO(int pages[],int s,int frame\_size){

    int page\_fault=0;

    int index=0;

    int arr[frame\_size]={-1};

    for(int i=0;i<frame\_size;i++){

        arr[i]=-1;

    }

    for(int i =0;i<s;i++){

        int a=pages[i];

        for(int i =0;i<frame\_size;i++){

            cout<< arr[i]<<" ";

        }

        cout<<endl;

        if(!findA(arr,a,frame\_size) ){

            page\_fault++;

            arr[index]=a;

            index++;

        }

        if(index==frame\_size){

            index=0;

        }

    }

    cout<<"page\_fault "<<page\_fault<<endl;

}

int main()

{

    int s;

    cout<<"enter size of page array"<<endl;

    cin>>s;

    int pages[s];

    cout<<"Enter array elements"<<endl;

    for(int i =0;i<s;i++){

        cin>>pages[i];

    }

    int frame\_size;

    cout<<"Enter the frame size"<<endl;

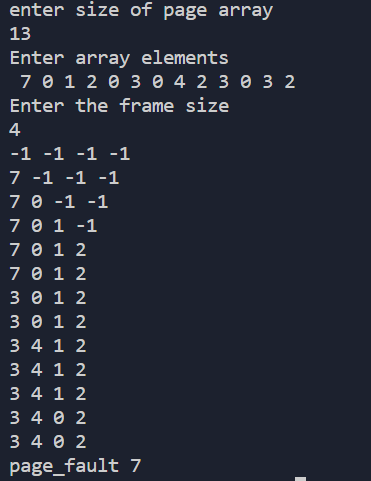
    cin>>frame\_size;

    FIFO(pages,s,frame\_size);

return 0;

}

**Output:**



**LRU**

#include<bits/stdc++.h>

using namespace std;

int pageFaults(int pages[], int n, int capacity){

    unordered\_set<int> s;

    unordered\_map<int, int> indexes;

    int page\_faults = 0;

    for (int i=0; i<n; i++){

        if (s.size() < capacity){

            if (s.find(pages[i])==s.end()){

                s.insert(pages[i]);

                page\_faults++;

            }

            indexes[pages[i]] = i;

        }

        else{

            if (s.find(pages[i]) == s.end()){

                int lru = INT\_MAX, val;

                for (auto it=s.begin(); it!=s.end(); it++){

                    if (indexes[\*it] < lru){

                        lru = indexes[\*it];

                        val = \*it;

                    }

                }

                s.erase(val);

                s.insert(pages[i]);

                page\_faults++;

            }

            indexes[pages[i]] = i;

        }

        for(auto i : s){

            cout<<i<<" ";

        }cout<<endl;

    }

    return page\_faults;

}

int main(){

    int n;

    cout<<"Enter the number of pages: ";

    cin>>n;

    int pages[n];

    cout<<"Enter the page references for each page: ";

    for(int i=0; i<n; i++){

        cin>>pages[i];

    }

    int capacity;

    cout<<"Enter the capacity: ";

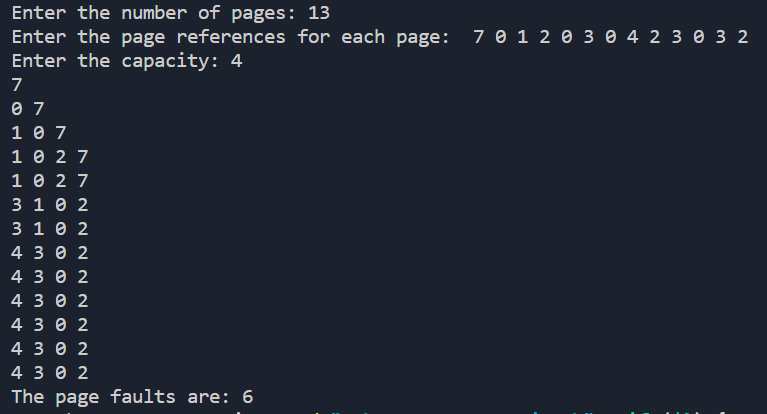
    cin>>capacity;

    cout<<"The page faults are: "<<pageFaults(pages, n, capacity);

    return 0;

}

**Output:**



**Optimal**

#include <bits/stdc++.h>

using namespace std;

bool search(int key, vector<int>& fr){

    for (int i = 0; i < fr.size(); i++)

        if (fr[i] == key)

            return true;

    return false;

}

int predict(int pg[], vector<int>& fr, int pn, int index){

    int res = -1, farthest = index;

    for (int i = 0; i < fr.size(); i++){

        int j;

        for (j = index; j < pn; j++){

            if (fr[i] == pg[j]){

                if (j > farthest){

                    farthest = j;

                    res = i;

                }

                break;

            }

        }

        if (j == pn)

            return i;

    }

    return (res == -1) ? 0 : res;

}

void optimalPage(int pg[], int pn, int fn){

    vector<int> fr(fn,-1);

    int hit = 0;

    for (int i = 0; i < pn; i++){

        if (search(pg[i], fr)){

            hit++;

            continue;

        }

        if (fr.size() < fn)

            fr.push\_back(pg[i]);

        else {

            int j = predict(pg, fr, pn, i + 1);

            fr[j] = pg[i];

        }

        for(int i =0;i<fn;i++){

            cout<<fr[i]<<" ";

        }

        cout<<endl;

    }

    cout<<"Number of hits: "<<hit<<endl;

    cout<<"Number of misses: "<<pn - hit<<endl;

}

int main(){

    int pn;

    cout<<"Enter the number of pages: ";

    cin>>pn;

    int pg[pn];

    cout<<"Enter the page references for each page: ";

    for(int i=0; i<pn; i++){

        cin>>pg[i];

    }

    int frames;

    cout<<"Enter the number of frames: ";

    cin>>frames;

    optimalPage(pg, pn, frames);

    return 0;

}

**Output:**

