**NOUSHAD AHMED(2K19/CO/266)**

**EXPERIMENT -7**

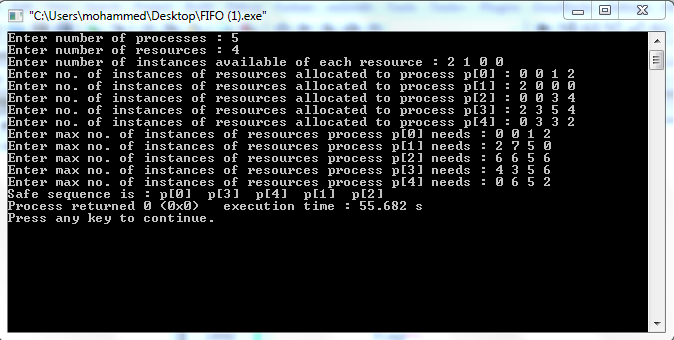
**AIM:**

To implement Banker’s Algorithm.

**Code:**

#include<iostream>  
using namespace std;  
class bankers\_alg  
{  
    int p,r,seq[20],al[20][20],rn[20][20],avl[20],ed;  
    int com\_pr();  
public:  
    void getdata();  
    void safe\_seq();  
};  
void bankers\_alg::getdata()  
{  
    cout<<"Enter number of processes : ";  
    cin>>p;  
    cout<<"Enter number of resources : ";  
    cin>>r;  
    cout<<"Enter number of instances available of each resource : ";  
    for(int i=0;i<r;i++)  
        cin>>avl[i];  
    for(int i=0;i<p;i++)  
    {  
        cout<<"Enter no. of instances of resources allocated to process p["<<i<<"] : ";  
        for(int y=0;y<r;y++)  
            cin>>al[i][y];  
    }  
    for(int i=0;i<p;i++)  
    {  
        cout<<"Enter max no. of instances of resources process p["<<i<<"] needs : ";  
        for(int y=0;y<r;y++)  
            cin>>rn[i][y];  
    }  
}  
int bankers\_alg::com\_pr()  
{  
    int flag=0,fl=0;  
    for(int i=0;i<p;i++)  
    {  
        for(int j=0;j<ed;j++)  
            if(i==seq[j])  
            {  
                fl=1;  
                break;  
            }  
        if(fl!=1)  
        {  
            for(int j=0;j<r;j++)  
            {  
                if(avl[j]-rn[i][j]+al[i][j]<0)  
                {  
                    flag=1;  
                    break;  
                }  
            }  
            if(flag==0)  
                return i;  
            flag=0;  
        }  
        fl=0;  
    }  
    return -1;  
}  
void bankers\_alg::safe\_seq()  
{  
    int temp,flag=0;  
    ed=0;  
    for(int i=0;i<p;i++)  
    {  
        temp=com\_pr();  
        if(temp!=-1)  
        {  
            for(int y=0;y<r;y++)  
                avl[y]+=al[temp][y];  
            seq[ed++]=temp;  
        }  
        else  
        {  
            cout<<"\nSystem is in unsafe state ";  
            flag=1;  
            break;  
        }  
    }  
    if(flag!=1)  
    {  
        cout<<"Safe sequence is : ";  
        for(int i=0;i<ed;i++)  
            cout<<"p["<<seq[i]<<"]  ";  
    }  
}  
int main()  
{  
    bankers\_alg banker;  
    banker.getdata();  
    banker.safe\_seq();  
    return 0;  
}

**OUTPUT SCREENSHOT:**



**LEARNING:**

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an 's-state' check to test for possible activities, before deciding whether allocation should be allowed to continue.

The Banker's algorithm got its name because it could be used in a banking system to ensure that the bank never allocated its available cash in such a way that it could no longer satisfy the needs of all its

customers.

**Properties -**

* Multiple instances
* Each process must a priori claim maximum use
* When a process requests a resource it may have to wait
* When a process gets all its resources it must return them in a finite amount of time