OS LAB 6

NAME: Aditya Anand

ROLL NO.: 20124009

BRANCH: IT

S NO.	TITLE	DATE OF IMPLEMENTATION	REMARKS
1	Simulation of Bankers Deadlock Avoidance and Prevention algorithms	23-02-2022	

SIMULATION OF BANKERS DEADLOCK AVOIDANCE AND PREVENTION ALGORITHM

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.

It checks if allocation of any resource will lead to deadlock or not, OR is it safe to allocate a resource to a process and if not then resource is not allocated to that process. Determining a safe sequence(even if there is only 1) will assure that system will not go into deadlock.

Banker's algorithm is generally used to find if a safe sequence exist or not.

CODE:

```
// C++ implementation of Banker's deadlock avoidance and prevention algorithm
#include<bits/stdc++.h>
using namespace :: std;

// Class representing state of the system: how various resources are allocated and requested by
various processes
struct state{
    int resources, processes;
    vector<vector<int> > allocated;
    vector<vector<int> > max_req;
    vector<vector<int> > remaining_req;
    vector<int> available;
    vector<bool> executed;

    state(int res, int pro){
        this->resources = res;
        this->processes = pro;

        allocated.resize(pro, vector<int>(res, 0));
        max_req.resize(pro, vector<int>(res, 0));
        remaining_req.resize(pro, vector<int>(res, 0));
        available.resize(res, 0);
        executed.resize(pro, false);
        cout<<"executed";
    }
};

queue<int> safe_seq;

bool bankers(state s){
    int n=s.processes;
    while(n--){
        cout<<"Current availability: ";
        for(int i=0; i<s.resources; i++){
```

```
            cout<<s.available[i]<<" ";
        }
        cout<<"\n";

        bool found=false;
        for(int i=0; i<s.processes; i++){
            bool t=true;
            if(!s.executed[i]){
                for(int j=0; j<s.resources; j++){
                    if(s.available[j]<s.remaining_req[i][j]){
                        t=false;
                        break;
                    }
                }
                if(t){
                    found=true;
                    safe_seq.push(i+1);
                    s.executed[i]=true;
                    cout<<"P"<<i+1<<" has been allocated the resources\n\n";
                    for(int j=0; j<s.resources; j++){
                        s.available[j]+=s.allocated[i][j];
                    }
                    break;
                }
            }
        }
        if(!found){
            cout<<"Resources could not be allocated to any process.\n\n";
            return true;
        }
    }
    return false;
}

int main(){
    cout<<"BANKER'S DEADLOCK AVOIDANCE AND PREVENTION ALGORITHM C++ IMPLEMENTATION\n";
    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";


    int res=0, pro=0;
    cout<<"Enter number of resources: ";
    cin>>res;
    cout<<"Enter number of processes: ";
    cin>>pro;

    state s(res, pro);

    cout<<"For "<<pro<<" processes enter: \n";
    for(int i=0; i<pro; i++){
        cout<<"Resources allocated to process "<<i+1<<": ";
        for(int j=0; j<res; j++){
            cin>>s.allocated[i][j];
```

```
        }
        cout<<"Max resources required by process "<<i+1<<": ";
        for(int j=0; j<res; j++){
            cin>>s.max_req[i][j];
            s.remaining_req[i][j]=s.max_req[i][j]-s.allocated[i][j];
        }
    }

    cout<<"Enter the currently available resources: ";
    for(int i=0; i<res; i++){
        cin>>s.available[i];
    }

    cout<<"\n\n-------------------------------------------------------------------------------
\n\n";
    bool deadlock = bankers(s);
    cout<<"\n\n-------------------------------------------------------------------------------
\n\n";

    if(deadlock){
        cout<<"Deadlock Situation Detected!\n";
    }
    else{
        cout<<"No Deadlock Detected!\n";
        cout<<"Safe sequence of resource allocation: ";
        while(!safe_seq.empty()){
            cout<<"P"<<safe_seq.front()<<" ";
            safe_seq.pop();
        }
        cout<<"\n";
    }
    return 0;
}
```

RESULT:

In case of Deadlock:

```
BANKER'S DEADLOCK AVOIDANCE AND PREVENTION ALGORITHM C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009        Branch: IT


Enter number of resources: 3
Enter number of processes: 3
executedFor 3 processes enter:
Resources allocated to process 1: 1 0 2
Max resources required by process 1: 3 1 2
Resources allocated to process 2: 4 3 2
Max resources required by process 2: 5 5 5
Resources allocated to process 3: 1 0 0
Max resources required by process 3: 1 1 0
Enter the currently available resources: 1 1 1


--------------------------------------------------------------------------------

Current availability: 1 1 1
P3 has been allocated the resources

Current availability: 2 1 1
P1 has been allocated the resources

Current availability: 3 1 3
Resources could not be allocated to any process.



--------------------------------------------------------------------------------

Deadlock Situation Detected!
```

No Deadlock:

```
BANKER'S DEADLOCK AVOIDANCE AND PREVENTION ALGORITHM C++ IMPLEMENTATION
Name: Aditya Anand      Roll No.:20124009        Branch: IT


Enter number of resources: 3
Enter number of processes: 3
executedFor 3 processes enter:
Resources allocated to process 1: 1 1 1
Max resources required by process 1: 2 3 2
Resources allocated to process 2: 1 0 0
Max resources required by process 2: 2 2 2
Resources allocated to process 3: 3 1 4
Max resources required by process 3: 5 1 5
Enter the currently available resources: 3 3 3


--------------------------------------------------------------------------------

Current availability: 3 3 3
P1 has been allocated the resources

Current availability: 4 4 4
P2 has been allocated the resources

Current availability: 5 4 4
P3 has been allocated the resources



--------------------------------------------------------------------------------

No Deadlock Detected!
Safe sequence of resource allocation: P1 P2 P3
```