OPERATING SYSTEM

Assignment-5

Name : prabhnoor
Roll No. : 102115059
3NC3

Simulate Bankers Algorithm for Dead Lock Avoidance and Prevention.

CODE:

```
#include <iostream>
#include <vector>
using namespace std;

const int MAX_PROCESSES = 10;
const int MAX_RESOURCES = 10;

int numProcesses, numResources;
vector<int> available(MAX_RESOURCES);
vector<vector<int>> maxClaim(MAX_PROCESSES,
vector<int>(MAX_RESOURCES)); vector<vector<int>>
allocation(MAX_PROCESSES, vector<int>(MAX_RESOURCES));
vector<vector<int>> need(MAX_PROCESSES, vector<int>(MAX_RESOURCES));

bool isSafe(vector<int>& work, vector<bool>& finish) {
    int count = 0; vector<int>
    safeSequence;

    while (count < numProcesses) {
        bool found = false;

        for (int i = 0; i < numProcesses; ++i) {
            if (!finish[i]) {
                bool canAllocate = true;

                for (int j = 0; j < numResources; ++j) {
                    if (need[i][j] > work[j]) {
                        canAllocate = false;
                        break;
                    }
                }

                if (canAllocate) {
                    for (int j = 0; j < numResources; ++j) { work[j]
                        += allocation[i][j];
```

```
                } finish[i] = true;
                safeSequence.push_back(i)
                ; count++;
                found = true;
            }
        }
    }

    if (!found) {
        return false;
    }
}

cout << "Safe sequence: "; for (int i = 0; i <
safeSequence.size(); i++) {
    cout << "P" << safeSequence[i];
    if (i < safeSequence.size() - 1) {
        cout << " -> ";
    }
}
cout << endl;

return true;
}

int main() {
    cout << "Enter the number of processes: "; cin
    >> numProcesses;

    cout << "Enter the number of resources: "; cin
    >> numResources;

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

    cout << "Enter the maximum claim matrix:" << endl; for
    (int i = 0; i < numProcesses; ++i) {
        for (int j = 0; j < numResources; ++j) {
            cin >> maxClaim[i][j];
        }
    }

    cout << "Enter the allocation matrix:" << endl; for
    (int i = 0; i < numProcesses; ++i) {
        for (int j = 0; j < numResources; ++j) {
            cin >> allocation[i][j];
            need[i][j] = maxClaim[i][j] - allocation[i][j];
        }
    }

    vector<int> work = available; vector<bool>
    finish(numProcesses, false);
```

```
    if (isSafe(work, finish)) {
        cout << "The system is in a safe state." << endl;
    } else { cout << "The system is in an unsafe state." <<
        endl;
    }

    return 0;
}
```

Output For Safe state example:

```
input
Enter the number of processes: 4
Enter the number of resources: 3
Enter the available resources: 3 3 2
Enter the maximum claim matrix:
7 5 3
3 2 2
9 0 2
2 2 2
Enter the allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
Safe sequence: P1 -> P3 -> P0 -> P2
The system is in a safe state.


...Program finished with exit code 0
Press ENTER to exit console.
```

Output For unsafe state example:

```
input
Enter the number of processes: 3
Enter the number of resources: 3
Enter the available resources: 1 2 2
Enter the maximum claim matrix:
1 2 3
3 1 4
5 6 7
Enter the allocation matrix:
0 1 1
1 2 1
3 4 6
The system is in an unsafe state.


...Program finished with exit code 0
Press ENTER to exit console.
```