Banker’s Algorithm

Code:

```
#include<iostream>
#include<queue>
#include<vector>

using namespace std;

int n, m;
vector<vector<int>> allocation;

vector<int> completeProcess(int process_number, vector<int>& available) {
    for (int i = 0; i < m; i++) {
        available[i] += allocation[process_number][i];
    }
    return available;
}

int main() {
    cout << "Enter number of processes: ";
    cin >> n;
    cout << "Enter number of resources: ";
    cin >> m;

    vector<vector<int>> max(n, vector<int>(m, 0)), need(n, vector<int>(m, 0));
    allocation.resize(n, vector<int>(m, 0));

    cout << "Enter the allocation instance for each process, resources" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> allocation[i][j];
        }
    }

    cout << "Enter the max values for each process, resources" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> max[i][j];
```

```
    }
}

cout << "Enter the initial available instances" << endl;
vector<int> available(m, 0);
for (int i = 0; i < m; i++) {
    cin >> available[i];
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        need[i][j] = max[i][j] - allocation[i][j];
    }
}

queue<pair<int, vector<int>>> process;
vector<int> result;

for (int i = 0; i < n; i++) {
    process.push({ i, need[i] });
}

int cnt = 0;

while (!process.empty()) {
    pair<int, vector<int>> current_process = process.front();
    process.pop();

    vector<int> temp = current_process.second;
    bool flag = true;

    for (int i = 0; i < m; i++) {
        if (temp[i] > available[i]) {
            flag = false;
            break;
        }
    }

    if (flag) {
        result.push_back(current_process.first);
        completeProcess(current_process.first, available);
        cnt = 0;
    } else {
        process.push(current_process);
```

```
            cnt++;
        }

        if (cnt == n) {
            cout << "Unsafe state" << endl;
            return 0;
        }
    }

    cout << "Safe state: ";
    for (int p : result) {
        cout << "P" << p << " ";
    }
    cout << endl;

    return 0;
}
```

Output: