TASKEEN SADIQ
DT-22004

OS LAB 08

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

using namespace std;

vector<vector<int>> maxMatrix, allocMatrix, needMatrix;
vector<int> avail;
int n, r;

void input();
void show();
void calculate();

int main() {
    cout << "********** Deadlock Detection Algorithm ************" << endl;
    input();
    show();
    calculate();
    return 0;
}

void input() {
    cout << "Enter the number of processes: ";
    cin >> n;
    cout << "Enter the number of resource instances: ";
    cin >> r;

    maxMatrix.resize(n, vector<int>(r));
    allocMatrix.resize(n, vector<int>(r));
    needMatrix.resize(n, vector<int>(r));
    avail.resize(r);

    cout << "Enter the Max Matrix:" << endl;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < r; j++)
            cin >> maxMatrix[i][j];

    cout << "Enter the Allocation Matrix:" << endl;

    for (int i = 0; i < n; i++)
```

TASKEEN SADIQ
DT-22004

```
        for (int j = 0; j < r; j++)
            cin >> allocMatrix[i][j];

    cout << "Enter the Available Resources:" << endl;
    for (int j = 0; j < r; j++)
        cin >> avail[j];
}

void show() {
    cout << "\nProcess\t Allocation\t Max\t Available\n";
    for (int i = 0; i < n; i++) {
        cout << "P" << i + 1 << "\t ";
        for (int j = 0; j < r; j++)
            cout << allocMatrix[i][j] << " ";

        cout << "\t ";
        for (int j = 0; j < r; j++)
            cout << maxMatrix[i][j] << " ";

        cout << "\t ";
        if (i == 0) {
            for (int j = 0; j < r; j++)
                cout << avail[j] << " ";
        }
        cout << endl;
    }
}

void calculate() {
    vector<int> finish(n, 0), deadProcesses;
    bool flag = true;

    // Compute Need Matrix
    for (int i = 0; i < n; i++)
        for (int j = 0; j < r; j++)
            needMatrix[i][j] = maxMatrix[i][j] - allocMatrix[i][j];

    while (flag) {
        flag = false;
        for (int i = 0; i < n; i++) {
            int count = 0;
            for (int j = 0; j < r; j++) {
```

TASKEEN SADIQ
DT-22004

```
            if (finish[i] == 0 && needMatrix[i][j] <= avail[j])
                count++;
        }

        if (count == r) {

        for (int j = 0; j < r; j++)
avail[j] += allocMatrix[i][j];

finish[i] = 1;
flag = true;
}
}
}

// Identify deadlocked processes
for (int i = 0; i < n; i++) {
if (finish[i] == 0)
deadProcesses.push_back(i + 1);
}

if (!deadProcesses.empty()) {
cout << "\nSystem is in Deadlock. The deadlocked processes are: ";
for (int process : deadProcesses)
cout << "P" << process << "\t";
cout << endl;
} else {
cout << "\nNo deadlock detected. System is in a safe state.\n";
}
}
```

TASKEEN SADIQ
DT-22004

```
********** Deadlock Detection Algorithm ************
Enter the number of processes: 3
Enter the number of resource instances: 3
Enter the Max Matrix:
3 2 2
2 2 2
2 1 2
Enter the Allocation Matrix:
1 1 0
1 0 1
0 0 1
Enter the Available Resources:
0 1 1

Process  Allocation      Max     Available
P1       1 1 0   3 2 2   0 1 1
P2       1 0 1   2 2 2
P3       0 0 1   2 1 2

System is in Deadlock. The deadlocked processes are: P1 P2      P3
```

```
********** Deadlock Detection Algorithm ************
Enter the number of processes: 3
Enter the number of resource instances: 3
Enter the Max Matrix:
3 2 2
2 1 2
2 2 2
Enter the Allocation Matrix:
1 1 0
1 0 1
1 1 1
Enter the Available Resources:
1 1 2

Process  Allocation      Max     Available
P1       1 1 0   3 2 2   1 1 2
P2       1 0 1   2 1 2
P3       1 1 1   2 2 2

No deadlock detected. System is in a safe state.
```