Deadlock Detection

Code:

```
#include <bits/stdc++.h>
using namespace std;
int arrmax[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;

void input()
{
    int i, j;
    cout << "Enter the no of Processes\t";
    cin >> n;
    cout << "Enter the no of resource instances\t";
    cin >> r;
    cout << "Enter the Max Matrix\n";
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            cin >> arrmax[i][j];
        }
    }
    cout << "Enter the Allocation Matrix\n";
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            cin >> alloc[i][j];
        }
    }
    cout << "Enter the available Resources\n";
    for (j = 0; j < r; j++)
    {
        cin >> avail[j];
    }
}
void show()
{
    int i, j;
    cout << "Process\t Allocation\t \tMax\t Available\t";
    for (i = 0; i < n; i++)
    {
        cout << "\nP" << i + 1 << "\t ";
        for (j = 0; j < r; j++)
        {
            cout << alloc[i][j] << " ";
        }
        cout << "\t\t";
        for (j = 0; j < r; j++)
        {
            cout << arrmax[i][j] << " ";
        }
        cout << "\t ";
        if (i == 0)
        {
            for (j = 0; j < r; j++)
                cout << avail[j] << " ";
        }
    }
}
void cal()
{
    int finish[100], temp, need[100][100], flag = 1, k, c1 = 0;
    int dead[100];
    int safe[100];
    int i, j;
    for (i = 0; i < n; i++)
    {
        finish[i] = 0;
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            need[i][j] = arrmax[i][j] - alloc[i][j];
        }
    }
    while (flag)
    {
        flag = 0;
        for (i = 0; i < n; i++)
        {
            int c = 0;
            for (j = 0; j < r; j++)
            {
                if ((finish[i] == 0) && (need[i][j] <= avail[j]))
                {
                    c++;
                    if (c == r)
                    {
                        for (k = 0; k < r; k++)
                        {
                            avail[k] += alloc[i][j];
                            finish[i] = 1;
                            flag = 1;
                        }
                        // cout<<"\nP%d",i;
                        if (finish[i] == 1)
                        {
                            i = n;
                        }
                    }
                }
            }
        }
    }
    j = 0;
    flag = 0;
    for (i = 0; i < n; i++)
    {
        if (finish[i] == 0)
        {
            dead[j] = i;
            j++;
            flag = 1;
        }
    }
    if (flag == 1)
    {
        cout << "\n\nSystem is in Deadlock and the Deadlock process 
are\n";
        for (i = 0; i < n; i++)
        {
            cout << "P" << dead[i] << "\t";
        }
    }
    else
    {
        cout << "\nNo Deadlock Occur";
    }
}
int main()
{
    int i, j;
    cout << "********** Deadlock Detection Algorithm ************\n";
    input();
    show();
    cal();
    return 0;
}
```

Output:

```
 *********** Deadlock Detection Algorithm ************
 Enter the no of Processes       5
 Enter the no of resource instances      4
 Enter the Max Matrix
 0 0 1 2
 1 7 5 0
 2 3 5 6
 0 6 5 2
 0 6 5 6
 Enter the Allocation Matrix
 0 0 1 2
 1 0 0 0
 1 3 5 4
 0 6 3 2
 0 0 1 4
 Enter the available Resources
 2 1 0 0
 Process  Allocation            Max      Available
 P1       0 0 1 2               0 0 1 2          2 1 0 0
 P2       1 0 0 0               1 7 5 0
 P3       1 3 5 4               2 3 5 6
 P4       0 6 3 2               0 6 5 2
 P5       0 0 1 4               0 6 5 6
 No Deadlock Occur
```

Deadlock Avoidance (Bankers)

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, m, i, j, k;
    cout << "Enter number of processes" << endl;
    cin >> n;
    cout << "Enter number of resources" << endl;
    cin >> m;
    int alloc[n][m];
    int max[n][m];
    int need[n][m];
    int av[m];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << "For process " << i + 1 << " Enter " << j + 1 <<
"allocated resource" << endl;
            cin >> alloc[i][j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << "For process " << i + 1 << " Enter " << j + 1 <<
"max resource" << endl;
            cin >> max[i][j];
        }
    }
    for (int i = 0; i < m; i++)
    {
        cout << "Enter resources" << endl;
        cin >> av[i];
    }
    i = 0;
    while (i < m)
    {
        int sum = 0;

        for (int j = 0; j < n; j++)
        {
            sum = sum + alloc[j][i];
        }
        av[i] = av[i] - sum;
        i++;
    }
    for (i = 0; i < n; i++)
    {

        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }
    int f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++)
    {
        f[k] = 0;
    }
    int y = 0;
```

```cpp
    for (k = 0; k < 10; k++)
    {
        for (i = 0; i < n; i++)
        {
            if (f[i] == 0)
            {

                int flag = 0;
                for (j = 0; j < m; j++)
                {
                    if (need[i][j] > av[j])
                    {
                        flag = 1;
                        break;
                    }
                }

                if (flag == 0)
                {
                    ans[ind++] = i;
                    for (y = 0; y < m; y++)
                        av[y] += alloc[i][y];
                    f[i] = 1;
                }
            }
        }
    }

    int flag = 1;
    for (int i = 0; i < n; i++)
    {
        if (f[i] == 0)
        {
            flag = 0;
            cout << "The given sequence is not safe";
            break;
        }
    }

    if (flag == 1)
    {
        cout << "Following is the SAFE Sequence" << endl;
        for (i = 0; i < n - 1; i++)
            cout << " P" << ans[i] + 1 << " ->";
        cout << " P" << ans[n - 1] + 1 << endl;
    }
    return 0;
}
```