```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>


int main()

{

    int n, m;

    printf("Enter the number of processes: ");

    scanf("%d", &n);


    printf("Enter the number of resource types: ");

    scanf("%d", &m);


    int max[n][m];

    int need[n][m];

    int allocation[n][m];

    int total[m]; // total resources of each type

    int aval[m];  // available resources

    int finish[n];

    int safeseq[n]; // safe sequence

    int count = 0;  // to track the safe sequence


    // Initialize the finish array to 0

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

    {

        finish[i] = 0;

    }


    // Input total resources of each type

    for (int i = 0; i < m; i++)

    {
```

```
    printf("Enter total number of resource type %d: ", i);

    scanf("%d", &total[i]);

}


// Input Max, Allocation, and calculate Need matrix

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

{

    printf("\nEnter details of process %d: \n", i);


    for (int j = 0; j < m; j++)

    {

        printf("Enter Max needs of resource %d for process %d: ", j, i);

        scanf("%d", &max[i][j]);

    }


    for (int j = 0; j < m; j++)

    {

        printf("Enter allocated resources of type %d to process %d: ", j, i);

        scanf("%d", &allocation[i][j]);

    }


    for (int j = 0; j < m; j++)

    {

        need[i][j] = max[i][j] - allocation[i][j]; // Calculate Need

    }


    printf("\n----------------------------------\n");

}


// Calculate available resources by subtracting allocated resources from total resources

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

```
{
    int sum_allocated = 0;
    for (int j = 0; j < n; j++)
    {
        sum_allocated += allocation[j][i];
    }
    aval[i] = total[i] - sum_allocated;
}

// Check if system is in a safe state
int executed_processes = 0;
while (executed_processes < n)
{
    bool found_process = false;

    for (int j = 0; j < n; j++)
    {
        if (finish[j] == 0)
        { // Process j hasn't finished
            bool can = true;
            for (int k = 0; k < m; k++)
            {
                if (need[j][k] > aval[k])
                { // If resources needed exceed available, can't execute
                    can = false;
                    break;
                }
            }
            if (can)
            { // If process j can be executed
                safeseq[count++] = j;
                finish[j] = 1; // Mark process j as finished
                executed_processes++;
                found_process = true;

                for (int k = 0; k < m; k++)
                {
                    aval[k] += allocation[j][k]; // Release resources
                }
            }
        }
    }

    // If no process was found in this iteration, deadlock exists
    if (found_process ==  false)

    {
        printf("Deadlock detected.\n");
        return 0;
    }
}

// If we reach this point, we found a safe sequence
printf("System is in a safe state. Safe sequence: ");
for (int i = 0; i < n; i++)
{
    printf("P%d ", safeseq[i]);
}
printf("\n");

return 0;
}
```