```c
#include <stdio.h>

int main() {
    int n, m;

    printf("Enter the no of processes :\n");
    scanf("%d", &n);

    printf("Enter the no of resources :\n");
    scanf("%d", &m);

    int max[n][m], alloc[n][m], avail[m];

    // Max Matrix Input
    printf("\nEnter the Max Matrix for each process :\n\n");
    for (int i = 0; i < n; i++) {
        printf("For process %d :\n", i + 1);
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
        printf("\n");
    }

    // Allocation Matrix Input
    printf("\nEnter the Allocation Matrix for each process :\n\n");
    for (int i = 0; i < n; i++) {
        printf("For process %d :\n", i + 1);
        for (int j = 0; j < m; j++) {
            scanf("%d", &alloc[i][j]);
        }
        printf("\n");
    }

    // Available Resources Input
    printf("Enter the Available Resources :\n");
    for (int j = 0; j < m; j++) {
        scanf("%d", &avail[j]);
    }

    // Print Max Matrix
    printf("\n\nMax matrix :\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            printf("%d ", max[i][j]);
        }
        printf("\n");
    }

    // Print Allocation Matrix
    printf("\nAllocation matrix :\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            printf("%d ", alloc[i][j]);
        }
        printf("\n");
    }

    // Banker's algorithm safety check
    int finish[n], need[n][m], safeFlag = 1;

    for (int i = 0; i < n; i++)
        finish[i] = 0;

    // Need = Max - Allocation
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
```

```c
            need[i][j] = max[i][j] - alloc[i][j];

    int completed = 0;

    while (completed < n) {
        int found = 0;

        for (int i = 0; i < n; i++) {
            if (!finish[i]) {
                int j;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j])
                        break;
                }

                if (j == m) {
                    for (int k = 0; k < m; k++)
                        avail[k] += alloc[i][k];

                    finish[i] = 1;
                    found = 1;
                    completed++;
                }
            }
        }

        if (!found) {
            safeFlag = 0;
            break;
        }
    }

    // Final Output
    printf("\nFinal output :\n");
    if (safeFlag)
        printf("The system is in a safe state!!\n");
    else
        printf("The system is in an unsafe state!!\n");

    return 0;
}
```