OS - Banker's Algorithm

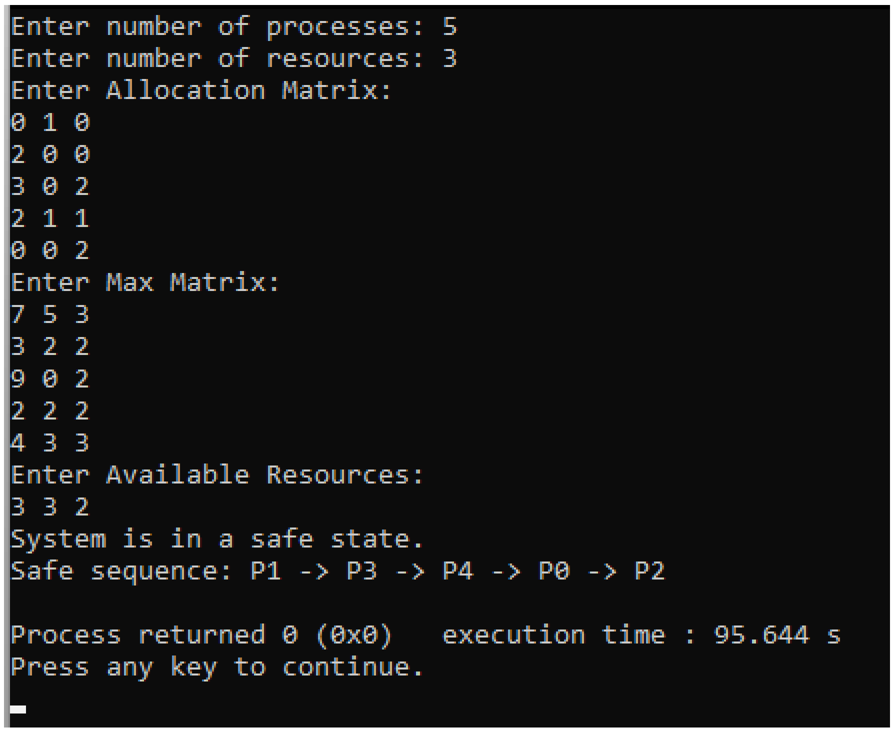
Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.

Code:

```
#include <stdio.h>
#include <stdbool.h>

#define MAX_PROCESSES 10
#define MAX_RESOURCES 10

int main() {
    int n, m; // n = number of processes, m = number of resources
    int allocation[MAX_PROCESSES][MAX_RESOURCES];
    int max[MAX_PROCESSES][MAX_RESOURCES];
    int need[MAX_PROCESSES][MAX_RESOURCES];
    int available[MAX_RESOURCES];
    bool finish[MAX_PROCESSES] = {false};
    int safeSequence[MAX_PROCESSES];

    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter number of resources: ");
    scanf("%d", &m);

    printf("Enter Allocation Matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            scanf("%d", &allocation[i][j]);

    printf("Enter Max Matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            scanf("%d", &max[i][j]);

    printf("Enter Available Resources:\n");
    for (int i = 0; i < m; i++)
    scanf("%d", &available[i]);

// Calculate Need matrix
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        need[i][j] = max[i][j] - allocation[i][j];

// Safety algorithm
int count = 0;
while (count < n) {
    bool found = false;
    for (int i = 0; i < n; i++) {
        if (!finish[i]) {
            bool canAllocate = true;
            for (int j = 0; j < m; j++) {
                if (need[i][j] > available[j]) {
                    canAllocate = false;
                    break;
                }
            }
            if (canAllocate) {
                for (int j = 0; j < m; j++)
                    available[j] += allocation[i][j];
                safeSequence[count++] = i;
                finish[i] = true;
                found = true;
            }
        }
    }
    if (!found) {
        printf("System is not in a safe state.\n");
        return 0;
    }
}

// Print Safe Sequence
printf("System is in a safe state.\nSafe sequence: ");
for (int i = 0; i < n; i++) {
    printf("P%d", safeSequence[i]);
```

```
            if (i != n - 1) printf(" -> ");
        }
        printf("\n");

        return 0;
    }
```

Output:



```
Enter number of processes: 5
Enter number of resources: 3
Enter Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available Resources:
3 3 2
System is in a safe state.
Safe sequence: P1 -> P3 -> P4 -> P0 -> P2

Process returned 0 (0x0)   execution time : 95.644 s
Press any key to continue.
```