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

**Program:**

#include <stdio.h>

#include <stdbool.h>

#define MAX\_PROCESSES 10

#define MAX\_RESOURCES 10

int main() {

int n, m;

int alloc[MAX\_PROCESSES][MAX\_RESOURCES];

int max[MAX\_PROCESSES][MAX\_RESOURCES];

int avail[MAX\_RESOURCES];

int need[MAX\_PROCESSES][MAX\_RESOURCES];

bool finished[MAX\_PROCESSES] = {false};

int safe\_sequence[MAX\_PROCESSES];

int count = 0;

printf("Enter number of processes and resources: ");

scanf("%d %d", &n, &m);

printf("Enter allocation matrix:\n");

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

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

scanf("%d", &alloc[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 matrix:\n");

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

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

}

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

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

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

}

}

printf("\nProcess\t\tAllocation\tMax\t\tNeed\n");

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

printf("P%d\t\t", i);

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

printf("%d ", alloc[i][j]);

}

printf("\t");

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

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

}

printf("\t");

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

printf("%d ", need[i][j]);

}

printf("\n");

}

while (count < n) {

bool found = false;

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

if (!finished[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];

}

safe\_sequence[count++] = i;

finished[i] = true;

found = true;

}

}

}

if (!found) {

printf("System is not in safe state.\n");

return 0;

}

}

printf("System is in safe state.\n");

printf("Safe sequence is: ");

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

printf("P%d", safe\_sequence[i]);

if (i != n - 1) {

printf(" -> ");

}

}

printf("\n");

return 0;

}

**Output:**

