15.Design a C program Illustrate the deadlock avoidance concept by simulating Banker's algorithm with C.

Program:

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

#define NUM_PROCESSES 5
#define NUM_RESOURCES 3

int available[NUM_RESOURCES];
int maximum[NUM_PROCESSES][NUM_RESOURCES];
int allocation[NUM_PROCESSES][NUM_RESOURCES];
int need[NUM_PROCESSES][NUM_RESOURCES];

bool isSafeState() {
   bool finished[NUM_PROCESSES] = {false};
   int work[NUM_RESOURCES];
   for (int i = 0; i < NUM_RESOURCES; i++) {
      work[i] = available[i];
   }

   for (int count = 0; count < NUM_PROCESSES; count++) {
      bool found = false;
      for (int i = 0; i < NUM_PROCESSES; i++) {
         if (!finished[i]) {
            bool canProceed = true;
            for (int j = 0; j < NUM_RESOURCES; j++) {
               if (need[i][j] > work[j]) {
                  canProceed = false;
                  break;
               }
            }
            if (canProceed) {
               for (int j = 0; j < NUM_RESOURCES; j++) {
                  work[j] += allocation[i][j];
               }
               finished[i] = true;
               found = true;
               break;
            }
         }
      }
      if (!found) {
```

```
            return false;
        }
    }
    return true;
}

int main() {
    // Input data
    printf("Enter the available resources:\n");
    for (int i = 0; i < NUM_RESOURCES; i++) {
        scanf("%d", &available[i]);
    }

    printf("Enter the maximum resources matrix:\n");
    for (int i = 0; i < NUM_PROCESSES; i++) {
        for (int j = 0; j < NUM_RESOURCES; j++) {
            scanf("%d", &maximum[i][j]);
        }
    }

    printf("Enter the allocation matrix:\n");
    for (int i = 0; i < NUM_PROCESSES; i++) {
        for (int j = 0; j < NUM_RESOURCES; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    // Calculate need matrix
    for (int i = 0; i < NUM_PROCESSES; i++) {
        for (int j = 0; j < NUM_RESOURCES; j++) {
            need[i][j] = maximum[i][j] - allocation[i][j];
        }
    }

    // Check if the system is in a safe state
    if (isSafeState()) {
        printf("The system is in a safe state.\n");
    } else {
        printf("The system is NOT in a safe state.\n");
    }

    return 0;
}
```

Output:

```
Enter the available resources:
3 3 2
Enter the maximum resources matrix:
7 5 3 3 2 2 9 0 2 2 2 2 4 3 3
Enter the allocation matrix:
0 1 0 2 0 0 3 0 2 2 1 1 0 0 2
The system is in a safe state.
```