OS Assignment 5 - Banker's Algorithm

Code :

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

#define MAX 10

// Function prototypes
void calculateNeed(int need[MAX][MAX], int max[MAX][MAX], int
allot[MAX][MAX], int p, int r);
bool isSafe(int processes[], int avail[], int max[][MAX], int
allot[][MAX], int p, int r);

int main() {
    int p, r;

    // Get the number of processes and resources from the user
    printf("Enter the number of processes: ");
    scanf("%d", &p);
    printf("Enter the number of resources: ");
    scanf("%d", &r);

    int processes[p], avail[r], max[p][r], allot[p][r];

    // Get the available resources
    printf("Enter the available resources for each resource type:\n");
    for (int i = 0; i < r; i++) {
        printf("Resource %d: ", i);
        scanf("%d", &avail[i]);
    }

    // Get the maximum demand matrix
    printf("Enter the maximum demand matrix:\n");
    for (int i = 0; i < p; i++) {
        printf("Process %d: ", i);
        for (int j = 0; j < r; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    // Get the allocation matrix
    printf("Enter the allocation matrix:\n");
    for (int i = 0; i < p; i++) {
        printf("Process %d: ", i);
        for (int j = 0; j < r; j++) {
            scanf("%d", &allot[i][j]);
        }
    }
```

```
    // Check system's safe state
    if (isSafe(processes, avail, max, allot, p, r)) {
        printf("System is in a safe state.\n");
    } else {
        printf("System is not in a safe state.\n");
    }

    return 0;
}

void calculateNeed(int need[MAX][MAX], int max[MAX][MAX], int
allot[MAX][MAX], int p, int r) {
    for (int i = 0 ; i < p ; i++) {
        for (int j = 0 ; j < r ; j++)
            need[i][j] = max[i][j] - allot[i][j];
    }
}

bool isSafe(int processes[], int avail[], int max[][MAX], int
allot[][MAX], int p, int r) {
    int need[MAX][MAX];
    calculateNeed(need, max, allot, p, r);

    bool finish[p];
    for (int i = 0 ; i < p ; i++)
        finish[i] = false;

    int safeSeq[p];
    int work[r];
    for (int i = 0 ; i < r ; i++)
        work[i] = avail[i];

    int count = 0;
    while (count < p) {
        bool found = false;
        for (int i = 0 ; i < p ; i++) {
            if (finish[i] == false) {
                int j;
                for (j = 0 ; j < r ; j++)
                    if (need[i][j] > work[j])
                        break;

                if (j == r) {
                    for (int k = 0 ; k < r ; k++)
                        work[k] += allot[i][k];

                    safeSeq[count++] = i;
                    finish[i] = true;
                    found = true;
                }
```

```
            }
        }

        if (found == false) {
            return false;
        }
    }

    printf("Safe sequence is: ");
    for (int i = 0; i < p ; i++)
        printf("%d ", safeSeq[i]);
    printf("\n");

    return true;
}
```

OUTPUT -