Banker's Algorithm Program in C

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>


int avail[100];

int total_res[100];

int total_alloc[100];

int req[100];


struct process {

    int pid;

    int max[100];

    int alloc[100];

    int need[100];

    int finish;

} p[20], temp;


void main() {

    int i, j, m, n, flag = 0, x = 0, k = 0, sequence[20], availtemp[100], a = 0, f = 0,

work[50];

    printf("ENTER THE NUMBER OF RESOURCES: ");

    scanf("%d", &m);
    printf("MAXIMUM RESOURCE COUNT FOR:
");
    for (j = 0; j < m; j++) {
        printf("\tRESOURCE %d: ", j);
        scanf("%d", &total_res[j]);
    }

    printf("\nENTER THE NUMBER OF PROCESSES: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        p[i].pid = i;
        printf("\nMAXIMUM ALLOCATION FOR PROCESS %d: ", p[i].pid);
        for (j = 0; j < m; j++)
            scanf("%d", &p[i].alloc[j]);

        printf("\nMAXIMUM REQUIREMENT FOR PROCESS %d: ", p[i].pid);
        for (j = 0; j < m; j++)
            scanf("%d", &p[i].max[j]);

        for (j = 0; j < m; j++)
            total_alloc[j] = total_alloc[j] + p[i].alloc[j];

        for (j = 0; j < m; j++)
            p[i].need[j] = p[i].max[j] - p[i].alloc[j];

        p[i].finish = 0;
```

```
}


printf("\n Matrix of Total Allocation\n");

for (i = 0; i < m; i++)

    printf("%d", total_alloc[i]);


printf("\n AVAILABLE MATRIX\n");

for (i = 0; i < m; i++)

    avail[i] = total_res[i] - total_alloc[i];


for (i = 0; i < m; i++) {

    work[i] = avail[i];

    printf("%d", avail[i]);

}


for (i = 0; i < (n - 1) * m; i++)

    availtemp[i] = 0;


while (a < m) {

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

        if (p[i].finish != 1) {

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

                if (work[j] >= p[i].need[j]) {

                    flag = 1;

                } else {

                    flag = 0;
                    break;
                }
            }

            if (flag != 0) {
                sequence[x] = p[i].pid;
                x++;
            }

            f = i * m;
            for (j = 0; j < m; j++) {
                if (flag != 0) {
                    work[j] += p[i].alloc[j];
                    availtemp[f] = work[j];
                    f++;
                    p[i].finish = 1;
                }
            }
        }
    }
    a++;
}

printf("\nPROCESS\tMAXIMUM\tALLOCATED\t NEED\t\tAVAIL\n");
for (i = 0; i < n; i++) {
    printf("\n%d\t", p[i].pid);
    printf("\t");

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

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

    }

    printf(" \t ");

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

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

    }

    printf(" \t ");

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

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

    }

    printf(" \t ");

    for (j = i * m; j < m * (i + 1); j++) {

        printf("%d ", availtemp[j]);

    }

    printf("\n");

}


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

    if (p[i].finish == 0)

        f = 1;

    else

        f = 0;

}
```

```
    if (f == 1)

        printf("\nSystem is Not Safe");

    else {

        printf("Sequence of Execution\n");

        for (i = 0; i < x; i++) {

            printf("P%d->", sequence[i]);

        }

        printf("\nSystem is Safe");

    }

}
```