LAB 6

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

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

void main() {
    int n, m, i, j, k;
    int alloc[10][10], max[10][10], need[10][10], avail[10];
    int f[10], ans[10], ind = 0;

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

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

    for (i = 0; i < n; i++) {
        printf("Enter allocation for P%d : ", i);
        for (j = 0; j < m; j++)
            scanf("%d", &alloc[i][j]);
        printf("Enter Max : ");
        for (j = 0; j < m; j++)
            scanf("%d", &max[i][j]);
    }

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

    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    for (k = 0; k < n; k++)
        f[k] = 0;

    printf("\n");

    for (k = 0; k < n; k++) {
        for (i = 0; i < n; i++) {
            if (f[i] == 0) {
                int flag = 0;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]) {
```

```
                    flag = 1;
                    break;
                }
            }
            if (flag == 0) {
                ans[ind++] = i;
                for (j = 0; j < m; j++)
                    avail[j] += alloc[i][j];
                f[i] = 1;
                printf("P%d is visited ( ", i);
                for (j = 0; j < m; j++)
                    printf("%d ", avail[j]);
                printf(")\n");
            }
        }
    }
}

int safe = 1;
for (i = 0; i < n; i++) {
    if (f[i] == 0) {
        safe = 0;
        printf("SYSTEM IS NOT IN SAFE STATE\n");
        break;
    }
}

if (safe) {
    printf("SYSTEM IS IN SAFE STATE\n");
    printf("The Safe Sequence is : ");
    for (i = 0; i < n - 1; i++)
        printf("P%d -> ", ans[i]);
    printf("P%d\n", ans[n - 1]);
}

printf("\nProcess    Allocation    Max    Need\n");
for (i = 0; i < n; i++) {
    printf("P%d\t\t", i);
    for (j = 0; j < m; j++)
        printf("%d ", alloc[i][j]);
    printf("\t");
    for (j = 0; j < m; j++)
        printf("%d ", max[i][j]);
    printf("\t");
    for (j = 0; j < m; j++)
        printf("%d ", need[i][j]);
    printf("\n");
}
```

```
}
```

OUTPUT:

```
PS D:\002> cd "d:\002\" ; if ($?) { gcc lab6.c -o lab6 } ; if ($?) { .\lab6 }
Enter number of processes : 5
Enter number of resources : 3
Enter allocation for P0 : 0 1 0
Enter Max : 7 5 3
Enter allocation for P1 : 2 0 0
Enter Max : 3 2 2
Enter allocation for P2 : 3 0 2
Enter Max : 9 0 2
Enter allocation for P3 : 2 1 1
Enter Max : 2 2 2
Enter allocation for P4 : 0 0 2
Enter Max : 4 3 3
Enter Available Resources : 3 3 2

P1 is visited (5 3 2 )
P3 is visited (7 4 3 )
P4 is visited (7 4 5 )
P0 is visited (7 5 5 )
P2 is visited (10 5 7 )
SYSTEM IS IN SAFE STATE
The Safe Sequence is : P1 -> P3 -> P4 -> P0 -> P2

Process    Allocation    Max    Need
P0             0 1 0   7 5 3   7 4 3
P1             2 0 0   3 2 2   1 2 2
P2             3 0 2   9 0 2   6 0 0
P3             2 1 1   2 2 2   0 1 1
P4             0 0 2   4 3 3   4 3 1
PS D:\002>
```