WEEK 6

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

CODE:

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

void main() {
    int alloc[10][10], max[10][10], avail[10], work[10];
    int need[10][10];
    char finish[10] = {0};
    int n, m;
    char safe_sequence[10][3];
    int count = 0;

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

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

    printf("Enter the maximum resource matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            scanf("%d", &max[i][j]);

    printf("Enter the available resource vector: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &avail[i]);
        work[i] = avail[i];
    }

    // Calculate the need matrix (need = max - alloc)
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
```

```
    // Safety Algorithm
    bool found = false;
    int index = 0;

    while (count < n) {
        found = false;
        for (int i = 0; i < n; i++) {
            if (!finish[i]) {
                bool can_execute = true;
                for (int j = 0; j < m; j++) {
                    if (need[i][j] > work[j]) {
                        can_execute = false;
                        break;
                    }
                }
                if (can_execute) {
                    for (int j = 0; j < m; j++)
                        work[j] += alloc[i][j];
                    finish[i] = 1;
                    sprintf(safe_sequence[index++], "P%d", i + 1);
                    count++;
                    found = true;
                }
            }
        }
        if (!found)
            break;
    }

    if (count == n) {
        printf("System is in a safe state.\nSafe sequence: ");
        for (int i = 0; i < n; i++) {
            printf("%s", safe_sequence[i]);
            if (i < n - 1)
                printf(" -> ");
        }
        printf("\n");
    } else {
        printf("System is not in a safe state.\n");
    }
}
```

OUTPUT:

```
Enter the number of processes:5
Enter the nuber of resources:3
Enter the allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 1
Enter the maximum resource matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the available resource vector: 3 3 2
System is in a safe state.
Safe sequence: P2 -> P4 -> P5 -> P1 -> P3
```

```
"C:\Users\Admin\Desktop\401\banker's a.exe"
Enter the number of processes:1
Enter the nuber of resources:3
Enter the allocation matrix:
3 0 2
Enter the maximum resource matrix:
7 5 3
Enter the available resource vector: 2 2 2
System is not in a safe state.
```