ABIRAMI K

230701008 CSE-A

EXP NO: 9 BANKERS DEADLOCK AVOIDANCE ALGORITHMS

PROGRAM

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

int main() {
    int n, m; // n = number of processes, m = number of resources
    printf("Enter the number of processes: ");
    scanf("%d", &n);

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

    int allocation[n][m], max[n][m], need[n][m], available[m];
    bool finish[n];
    int safeSeq[n];

    // Input Allocation Matrix
    printf("Enter Allocation Matrix:\n");
    for (int i = 0; i < n; i++) {
        printf("P%d: ", i);
        for (int j = 0; j < m; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    // Input Max Matrix
    printf("Enter Max Matrix:\n");
    for (int i = 0; i < n; i++) {
        printf("P%d: ", i);
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
            need[i][j] = max[i][j] - allocation[i][j]; // Calculating Need Matrix
        }
    }

    // Input Available Resources
    printf("Enter Available Resources:\n");
    for (int i = 0; i < m; i++) {
        scanf("%d", &available[i]);
    }

    // Banker's Algorithm
    int count = 0;
    for (int i = 0; i < n; i++) finish[i] = false;
```

```
    // Input Available Resources
    printf("Enter Available Resources:\n");
    for (int i = 0; i < m; i++) {
        scanf("%d", &available[i]);
    }

    // Banker's Algorithm
    int count = 0;
    for (int i = 0; i < n; i++) finish[i] = false;

    while (count < n) {
        bool found = false;
        for (int i = 0; i < n; i++) {
            if (!finish[i]) {
                bool canAllocate = true;
                for (int j = 0; j < m; j++) {
                    if (need[i][j] > available[j]) {
                        canAllocate = false;
                        break;
                    }
                }

                if (canAllocate) {
                    for (int j = 0; j < m; j++)
                        available[j] += allocation[i][j];

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

        if (!found) {
            printf("\nSystem is NOT in a safe state.\n");
            return 0;
        }
    }

    // If we reach here, system is in safe state
    printf("\nThe SAFE Sequence is:\n");
    for (int i = 0; i < n; i++) {
        printf("P%d", safeSeq[i]);
        if (i != n - 1) printf(" -> ");
    }
    printf("\n");

    return 0;
```

OUTPUT

```
Enter the number of processes: 5
Enter the number of resources: 3
Enter Allocation Matrix:
P0: 0 1 0
P1: 2 0 0
P2: 3 0 2
P3: 2 1 1
P4: 0 0 2
Enter Max Matrix:
P0: 7 5 3
P1: 3 2 2
P2: 9 0 2
P3: 2 2 2
P4: 4 3 3
Enter Available Resources:
3 3 2

The SAFE Sequence is:
P1 -> P3 -> P4 -> P0 -> P2
```