

OS LAB MANUAL (CS23431)

Roll No:230701258

EX.NO:9

DEADLOCK AVOIDANCE

Aim: To find out a safe sequence using Banker's algorithm for deadlock avoidance.

Program:

```
#include <stdio.h>
#include <stdbool.h>
int main() {
    int n, m, i, j, k;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the number of resources: ");
    scanf("%d", &m);
    int alloc[n][m], max[n][m], avail[m], need[n][m];
    bool finish[n];
    int safeSeq[n];
    printf("\nEnter the Allocation Matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            scanf("%d", &alloc[i][j]);
    printf("\nEnter the Maximum Matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            scanf("%d", &max[i][j]);
    printf("\nEnter the Available Resources:\n");
    for (i = 0; i < m; i++)
        scanf("%d", &avail[i]);
    for (i = 0; i < n; i++) {
        finish[i] = false;
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
```

```

    }

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

if (count == n) {
    printf("\nThe SAFE Sequence is:\n");
    for (i = 0; i < n; i++) {
        printf("P%d", safeSeq[i]);
        if (i != n - 1) printf(" -> ");
    }
    printf("\n");
}

```

```
    } else {  
        printf("\nThere is NO SAFE SEQUENCE. The system is in an unsafe  
state.\n");  
    }  
    return 0;  
}
```

Input:

```
Enter the number of processes: 4  
Enter the number of resources: 2  
  
Enter the Allocation Matrix:  
1  
2  
3  
4  
5  
6  
7  
8  
  
Enter the Maximum Matrix:  
4  
6  
8  
7  
9  
5  
6  
5  
  
Enter the Available Resources:  
6  
5
```

Output:

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