OS LAB MANUAL (CS23431)

Roll No:230701234

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], safeSeq[n];
    bool finish[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];
for (i = 0; i < m; i++)
    work[i] = avail[i];

int count = 0;

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;
```

```
                    break;
                }
            }
        }

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

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

    return 0;
}
```

Input:

```
pranav@Pranav:~$ vi NINE.c
pranav@Pranav:~$ gcc NINE.c
pranav@Pranav:~$ ./a.out
Enter number of processes: 5
Enter number of resource types: 3

Enter the Allocation Matrix:
For Process 0: 0 1 0
For Process 1: 2 0 0
For Process 2: 3 0 2
For Process 3: 2 1 1
For Process 4: 0 0 2

Enter the Maximum Matrix:
For Process 0: 7 5 3
For Process 1: 3 2 2
For Process 2: 9 0 2
For Process 3: 2 2 2
For Process 4: 4 3 3

Enter the Available Resources:
Resource 0: 3
Resource 1: 3
Resource 2: 2
```

Output:

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