Ex. No.: 9
Date:

## DEADLOCK AVOIDANCE

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

**Algorithm:**
1. Initialize work=available and finish[i]=false for all values of i
2. Find an i such that both:
finish[i]=false and Need$_i$<= work
3. If no such i exists go to step 6
4. Compute work=work+allocationi
5. Assign finish[i] to true and go to step 2
6. If finish[i]==true for all i, then print safe sequence
7. Else print there is no safe sequence

**Program Code:**

```
#include<stdio.h>
int main()
{
    int p, r, count=0, i, j, alc[5][3], max[5]
        need[5][3], safe[5], available[3],
        done[5], terminate=0;
    printf("Enter the number of process and
            resources");
    scanf("%d %d", &p, &r);
    printf("Enter allocation of resource
            of all process %d x %d matrix",
            p, r);

    for(i=0; i<p; i++){
        for(j=0; j<r; j++){
            scanf("%d", &alc[i][j]);
```

}
}

```
printf("Enter the maximum number of
        processes: "); scanf("%d", &p);

for(i=0; i<p; i++){
    for(j=0; j<r; j++)
        scanf("%d", &max[i][j]);
}

printf("Enter the available number");
for(i=0; i<r; i++)
    scanf("%d", &available[i]);
printf("\n\n Now enter the matrix one by");

for(i=0; i<p; i++){
    for(j=0; j<r; j++){
        need[i][j] = max[i][j] -
                     alloc[i][j];
        printf("%d\t", need[i][j]);
    }
    printf("\n");
}

for(i=0; i<p; i++)
    done[i] = 0;

while(count < p){
```

```
for(i=0; i<p; i++){
    if(done[i]==0){
        for(j=0; j<c; j++){
            if(need[i][j] >
                available[j]
            break;
        }
        if(j==c){
            safe[count] = i;
            done[i] = 1;
            for(j=0; j<c; j++){
                available[j]
                += alc[i][j]
            }
            count++;
            terminate=0;
        }
        else{
            terminate++;
        }
    }
}
```

Output :-

Enter no. of process & resource : 5 3

Enter alloc of resource of all process : 0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter max resource required :
7 5 3
3 2 2
9 0 2
4 0 2
5 8 3

Enter available process : 3 3 2

Need resource matrix : 7 4 3
1 2 2
6 0 0
2 1 1
5 3 1

Available resource after completion : 10 5 7

Safe sequence are : $P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$

**Sample Output:**

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

**Result:**

Hence safe sequence is obtained for deadlock avoidance using banker's algo.