Ex. No.: 9
Date: 23.4.24

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+$allocation_i$
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:

```
// vi bankers.c
#include<stdio.h>
int main(){
        int n,m,i,j,k;
        n=5, m=3;
        int alloc[5][3]={{0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2}};
        int max[5][3]={{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3}};
        int avail[3]= {3,3,2};
        int f[n], ans[n], ind=0;
        for(k=0;k<n;k++){
                f[k]=0;
        }
        int need[n][m];
        for(i=0;i<n;i++){
                for(j=0;j<m;j++){
                        need[i][j]=max[i][j]-alloc[i][j];
        }
int y=0;
```

```
for (k=0 ; k<5 ; k++){
        for (i=0 ; i<n ; i++) {
            if(f[i]==0){
                int flag=0 ;
                for(j=0 ; j<m ; j++){
                    if( need[i][j] > avail[j]) {
                        flag=1 ;
                        break ;
                    }
                }
                if(flag==0){
                    ans[ind++]=i ;
                    for(y=0 ; y<m ; y++)
                        avail[y] += alloc[i][y] ;
                    f[i]=1 ;
                }
            }
        }
}

printf(" The SAFE Sequence is \n");
for(i=0 ; i<n-1 ; i++)
        printf(" P%d -> ", ans[i]) ;

printf(" P%d \n ", ans[n-1]);
return 0 ;
}
```

Output: gcc bankers.c
./a.out

The SAFE sequence is

P1 → P3 → P4 → P0 → P2

23/4/24

RESULT:

The program has been compiled & executed and the output has been verified successfully.