Ex. No.: 9
Date: 3/4/25

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 Needᵢ<= 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>
# include< stdbool.h>
# define MAX_PROCESSES 5
# define MAX_RESOURCES 3
bool is safe (int process[ ], int avoid[], int max[], [MAX_PROCESSES],
        int allot[][MAX_PROCESSES][MAX_RESOURCES];
        bool finish[MAX_PROCESSES] = [false];
        int work[MAX_RESOURCES];
        for (int i=0; i<n; i++){
            for (int j=0; j<m; j++){
                need[i][j] = max[i][j] - allot[i][j];
            }
        }
        for (int i=0; i<m; i++){
            work[i]= avoid[i];
        }
        int safe sequence[MAX_PROCESSES];
        int count=0;
```

```
while (count < n){
    bool found = false;
    for (int i=0; i<n; i++){
        if (!finished[i]){
            int j;
            for (j=0; j<m; j++){
                if (need[i][j] > work[j])
                    break;
            }
            if (j==m){
                for (int k=0; k<m; k++){
                    work[k]+ = allot[i][k];

            }
            finish[i] = true;
            Safe sequence[count++]=i;
            found = true;
            break;
            }
        }
    }
    printf(" Safe Sequence");
    for (int i=0; i<n; i++){
        print ("p %d", safe sequence[i]);
    }
    printf("\n");
    return true;
}
int main(){
    int process[MAX_PROCESSES];
    int avail[MAX_RESOURCES];
    printf ("Enter available resources (ABC):");
    for(i=0; i<MAX_RESOURCES; i++){
        scanf(" %d", & avail[i]);
    }
```

```
int max[MAX_PROCESSES][MAX_RESOURCES];
printf("Enter nax demand metrix/. Max resources per each process \n");
for(int i=0; i<MAX_PROCESSES ;i++){
        printf("Enter max demand matrix\. ("Max resources per each process):\n");
for(int i=0;i<MAX_PROCESSES;i++){
        printf("Enter max demand for process ")

if(!safe(process, avail, max, allo', n, m)
            return0;
return0;
}
```

OUTPUT

```
Enter no of procss & resouro : 5  3
Enter alloc of resources of all procss.   0 1 0
                                          2 0 0
                                          3 0 2
                                          3 1 1
                                          0 0 2

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

Enter Available resource : 332
Need Resource Matrix : 7 4 3
                       1 2 2
                       6 0 0
                       2 1 1
                       5 3 1
```

Sample Output:

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

```
Available resource afte completion : 10 5 7
Safe seq ance : P1 -> P3 -> P4 -> PC -> P2
```

Result:

Hence safe sequence is oblaine for dead lock Avoidance using bankers algorithm.