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 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, c, count = 0, i, j, alc[5][3], max[5][3], need[5][3], safe[5],
    available[3], done[5], terminate = 0;
    printf("Enter the number of procs and resources");
    scanf("%d %d", &p, &c);
    printf("enter allocation of resources of all procs %d x %d matrix", p, c);
    for(i=0; i<p; i++) {
        for(j=0; j<c; j++) {
            scanf("%d", &alc[i][j]);
        }
    }
    printf("enter the max resource procs required %d x %d matrix", p, c);
    for(i=0; i<p; i++) {
        for(j=0; j<c; j++) {
            scanf("%d", &max[i][j]);
        }
    }
    printf("enter the available resources");
    for(i=0; i<c; i++)
        scanf("%d", &available[i]);
    printf("\n need resources matrix are\n");
```

```
for(i=0 ; i<p; i++) {
    for(j=0 ; j<c ; j++) {
        need [i][j] = max[i][j] - alc [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++;
            }
        }
    }
    if (terminate == (p-1)) {
        printf (" safe sequne does not
                exist");
        break;
    }
    if (terminate != (p-1)) {
        printf ("\n available resource after 57
                completion \n");
        for (i = 0; i < c ; i++) {
            printf ("%d \t", available[i]);
        }
        printf ("\n safe sequence are \n");
        for (i=0 ; i<p ; i++) {
            printf (" p%d \t", safe[i]);
        }
    }
}
return 0;
}
```

Sample Output:

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

OUTPUT

5 3

allocation	max	available
0 1 0	7 5 3	3 3 2
2 0 0	3 2 2	
3 0 2	9 0 2	
2 1 1	4 2 2	
0 0 2	5 3 3	

need

7 4 3
1 2 2
6 0 0
2 1 1
5 3 1

available resource

10 5 7

safe sequence

$< P_1, P_3, P_4, P_0, P_2 >$

Result:

Thus the above code for deadlock avoidance using bankers algorithm is successfully executed