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

# 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>
#include <stdbool.h>

int main() {
    int ~~max~~ n,m;
    printf(" Enter number of process:");
    scanf(" %d", &n);
    printf(" Enter number of resources:");
    scanf("%d", &m);
    int max[n][m];
    printf(" Enter the value for max array:");
        for ( int i=0; i<n; i++) {
            for (int J=0; J<m; J++) {
                scanf(" %d", &max[i][J]);
            }
        }
    int allocate [n][m];
    printf(" Enter values for allocate array:");
```

```
for ( int i=0; i<n ; i++) {
        for (int J=0; J<m ; J++) {
        Scanf ("%d", & allocate [i] [J]);
    }
}
int  avail [m] ;
  for ( int i=0 ; i<m ; i++)
  {
    printf (" Enter Avail [%d]", i);
    Scanf ("%d", & avail [i]);
  }
  int  Need [n] [m];
   for (int  i=0; i<n; i++) {
                  for ( int  J=0 ; J<m ; J++)
             {
              Need [i] [J]  = max [i] [J] - allocate [i] [J];
             }
     }
 int  work [m];
 boolean finish [n];
  for  (int  i=0 ; i<m ; i++)
  {
       work [i]  = avail [i] ;
  }
 for (int i=0 ; i<n ; i++)
           finish [i]  = false ;
  int  Sef [n];
  int ~~flag~~ ; ind = 0;
  While  (ind != n) {
```

Max Array

$$\begin{bmatrix} 7 & 5 & 3 \\ 3 & 2 & 2 \\ 9 & 0 & 2 \\ 2 & 2 & 2 \\ 4 & 3 & 3 \end{bmatrix}$$

Allocation Array

$$\begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Available Array

$[3 \quad 2 \quad 2]$

Need Array

$$\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

Safe sequence :

$P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$

```
for (int i=0 ; i<n ; i++) {

      flag =1 ;
   if ( finished [i]  == false ) {
         for ( int J =0 ;  J<m ; J ++) {
         if ( Need [i][J] > work [J])
           flag = 0;
     }
    if ( flag == 1) {
        for ( int J=0 ; J <m ; J++) {
            finish [i] = true;
           work [J] += allocate [i][J];
          }
          seq [ind ++] = i;
        }
      }
    }
  }

printf ("The SAFE SEQUENCE is \n");
 for (int i=0 ; i<n -1 ; i ++)
     printf ("p%d → ", seq [i]);
   printf ("p%d", seq [ind -1]);
}
```

**Sample Output:**

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

**Result:**

Program to find out a safe sequence using banker's algorithm for deadlock was written & executed successfully.

[illegible]