**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 n,m;
  printf("Enter number of processes:");
  scanf("%d",&n);
  printf("Enter number of resources:");
  scanf("%d",&m);
int allocation[n][m],max[n][m],need[n][m],available[m];
printf("Enter allocation matrix:\n");
  for(int i=0;i<n;i++)
    for(int j=0;j<m;j++)
    scanf("%d",&max[i][j]);
  printf("Enter available resources:\n");
  for(int i=0;i<n;i++)
    for(int j=0;j<m;j++)
    scanf("%d",&max[i][j]);
```

```
printf("Enter available resources:\n");
for(int j = 0; j<m; j++)
  scanf("%d", &available[0][j]);

for(int i=0; i<n; i++)
  for(int j=0; j<m; j++)
    need[i][j] = max[i][j] - allocation[i][j];

int finish[n];
for(int i=0; i<n; i++)
  finish[i]=0;

int SafeSequence[n], count=0;
  int Work[m];
for(int i=0; i<m; i++)
  work[i] = available[i];
while(count<n){
  int found=0;
for(int i=0; i<n; i++){
  if(finish[i]==0){
  int canRun=1;
for(int j=0; j<m; j++){
  if(need[i][j]>work[j]){
    canRun=0; //
    break;
  }
}
```

```
if(Can Run){
  for(int j=0;j<m;j++)
  work[j]+=allocation[i][j];

  Safe Sequence[cnt++]=i;

  finish[i]=1;
  fish found=1;

    }
  }
}
if(found){
 printf("System is NOT in a Safe state. No Safe Sequence.\n");
  return 0;

}
}
printf("System is in a Safe state. In Safe Sequence:\n");
   for(int i=0;i<n;i++)
     printf("P-%d", Safe Sequence[i]);
   printf("\n");
   return 0;
}
```

Input:

Enter number of processes : 5
Enter number of resources : 3

Enter allocation matrix

0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter max matrix

7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Enter available resources

3 3 2

**Sample Output:**

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

Output:

System is in a safe state

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

**Result:**

Thus, safe sequence using Banker's algorithm for deadlock detection executed successfully.