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 m, n;
    Printf (" Enter the no of resources & processes\n");
    Scanf("%d %d ", &m , &n);
     int max[n][m], allocation[n][m];
     Printf("\n Enter the values for max array:\n");
     for(int i=0; i<n ; i++)
     { for(int j=0; j<m; j++)
       { scanf( "%d",&max[i][j]);
       }
     }
     printf("\n Enter the values for allocation array :\n");
     for(int i=0; i<n; i++)
     {
       for(int j=0; j<n; j++)
```

```
{allocation[
  scanf("%d", &allocation[i][j]);
}}
printf("Enter the value for available array:\n");
int avail[m];
for(int i=0; i<m; i++)
{
  scanf("%d", &avail[i]);
}
int need[n][m];
printf("Need Matrix\n");
for(int i=0; i<n; i++)
{ for(int j=0; j<m; j++)
  {
    need[i][j]= max[i][j]-allocation[i][j];
    printf("%d ", need[i][j]);
  }
  printf("\n");
}
int work[m];
for(int i=0; i<m; i++)
{
  work[i] = avail[i];
}
bool finish[n];
for(int i=0; i<n; i++)
{
  finish[i]=false;
}
int s[n], k=0;
while(k<n)
{ for(int i=0; i<n; i++)
  { if(!finish[i])
```

```
{
    bool f = true;
    for(int j=0 ; j<m ; j++)
    {
        if(need[i][j] > work[j]){
                f = false;
                break;
        }
    }
    if(f)
    {
        for(int j=0 ; j<m; j++)
        {
            work[j] += allocation[i][j];
        }
        finish[i] = true;
        s[k] = i;
        k++;
    }
}
}
printf("The Safe sequence is : \n");
for(int i=0; i<n-1; i++)
{
    printf("p%d →", s[i]);
}
printf("p%d", s[n-1]);
}
```

Max

	A	B	C
P_0	7	5	3
P_1	3	2	2
P_2	9	0	2
P_3	2	2	2
P_4	4	3	3

Allocation

P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need

	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Safe Sequence

$\langle P_1 \; P_3 \; P_4 \; P_0 \; P_2 \rangle$

Input:

Enter the no of resources & processes

3

5

Enter value for max array:

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter value for allocation array:

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

Enter value for available array:

3 3 2

Sample Output:

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

O/P

Need matrix

7 4 3

1 2 2

6 0 0

0 1 1

4 3 1

The Safe sequence is:

P1 → P3 → P4 → P0 → P2.

Result:

Thus the C program for Deadlock avoidance is sucessfully executed.