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, r, i, j, k;
  n = 3;
  r = 3;
  int alloc [3] [3] = {{0,1,1}, {0,1,0}, {1,1,2}};
  int max [3] [3] = {{4,3,0}, {5,4,1}, {6,5,2}};
  int avail [3] = {0,1,0};
  int f[n], ans[n], ind = 0;
  for (k=0; k<n; k++)
  {  f[k]=0;
  }
  int need [n] [r];
  for (i=0; i<n; i++)
  {
  for (j=0; j<r; j++)
  {  need[i][j] = max[i][j] - alloc[i][j];
  }
  }
int y=0;
infor (k=0; k<n; k++)
 {
   for (i=0; i<n; i++)
   {
      if (f[i]==0)
      {
        int flag=0;
        for (j=0; j<r; j++)
        {
          if (need [i][j] > avail[j])
          {
            flag=1;
            break
          }
        }
        if (flag==0)
        {
          ans[ind ++]=i;
          for (y=0; y<r; j++)
             avail[y]+= alloc [i][y];
        f[i]=1;
      }found=1;
   } }
 }
 ~~printf ("The SAFE Sequence is: ");~~
 ~~for (i=0; i<n-1; i++)~~
 if (!found)
 {  safe =0;
    break;
 }
}
if (safe)
{ printf ("The Safe Sequence is: ");
  for (int i=0; i<n-1; i++)
  {  printf ("P%d\n", ans[n-1]);
  }
  else
   printf ("The system is Not in a Safe state.");
}
return 0;
}
```

Sample Output:

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

The Safe Sequence is:
P2 → P1 → P0

Result:

Hence the Deadlock Avoidance using Banker's Algorithm is implemented and executed

[illegible]