Ex. No.: 9
Date: 5/4/25

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:

```
allocated = [[0,1,0],[2,0,0],[3,0,2], [2,1,1],[0,0,2]]
max_matrix = [[7,5,3],[3,2,2],[9,0,2], [2,2,2],[4,3,3]]
need = []
print("Need Matrix")
for i in range(5):
        l=[]
        print()
        for j in range (3):
                l.append (max_matrix [i][j]-allocated[i][j])
                print(l[j],end=" ")
        need.append (l)
print()
ss=[]
available =[3,3,2]
finish = [False, False, False, False, False]
j=5
i=0
while (j!=0):
```

```
if (finish[i]== False):
    if (available[0] >=need[i][0] and available[1] >=need[i][1]an
    available [2] >=need [i][2]):
            finish[i]= True
            j-=1.
            available [0]+= allocated [i][0]
            available [1]+= allocated [i][1]
            available [2]+=allocated [i][2]
            ss.append(i)
    i=(i+1)%5
print("The safe sequence is ")
for i in ss:
    print ("P"+str(i), end="->")
```


PDF Scanner ACE Scanner

Sample Output:

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

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 program to implement solution to find out safe sequence using Banker's algorithm for deadlock avoidance has been executed successfully

PDF Scanner ACE Scanner