Ex. No.: 9
Date: 03/04/2025

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:

```
def bankers_algorithm(processes, available, max_need,
                      allocation):

    n = len(processes)
    m = len(available)
# calculating need
    need = []
    for i in range(n):
        need_row = []
        for j in range(m):
            need_row.append(max_need[i][j] - allocation
                                              [i][j])
        need.append(need_row)

# initialising
    finish = [0]*n
    work = available[:]
    safe_sequence = []
```

```
# Checking
while len(safe_sequence) < n:
    progress = False
    for i in range(n):
        if finish[i] == 0:
            can_allocate = True
            for j in range(m):
                if need[i][j] > work[j]:
                    can_allocate = False
                    break
            if can_allocate:
                for j in range(m):
                    work[j] += allocation[i][j]
                finish[i] = 1
                safe_sequence.append(processes[i])
                progress = True
    if not progress:
        break
if len(safe_sequence) == n:
    print("Safe sequence found:")
    for p in safe_sequence:
        print(p, end=" ")
    print()
else:
    print("No safe sequence found. System is not in a safe state.")
```

Output:

Enter no. of Processes: 3
Enter no. of resource types: 3
Enter process names:
P_1
P_2
P_3
Enter available resources:
3 3 2
Enter allocation matrix:
0 1 0
2 0 0
3 0 2

Sample Output:

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

Enter max matrix:
7 5 3
3 2 2
9 0 2
No safe sequence found. System is not in a safe state.

Result:

Thus the Python program for deadlock avoidance using Banker's algorithm has been executed successfully.