Ex. No. : 9 **Date : 05.04.2025**

Register No. : 230701385 **Name : VISHWAK S**

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+allocation
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:

```
def bankers_algo(process, available, max, allocation):
    n_process = len(process)
    n_resource = len(available)
    need = []
    safe_sequence = []

    for i in range(n_process):
        process_need = []
        for j in range(n_resource):
            process_need.append(max[i][j] - allocation[i][j])
        need.append(process_need)

    print("Need Matrix")
    for i in range(n_process):
        print(f"{process[i]}        ", end="")
        for j in range(n_resource):
            print(f"{need[i][j]}   ", end="")
        print()

    work = available.copy()
    finish = [False] * n_process
    count = 0

    while count < n_process:
        found = False
        for i in range(n_process):
            if finish[i]==False:
                can_allocate = all(need[i][j] <= work[j] for j in range(n_resource))
                if can_allocate:
                    for j in range(n_resource):
                        work[j] += allocation[i][j]
                    safe_sequence.append(process[i])
                    finish[i] = True
                    count += 1
                    found = True
        if not found:
            break

    if count == n_process:
        return safe_sequence
    else:
        return []
```

```python
def main():
    process = ["P0", "P1", "P2", "P3", "P4"]
    available = [1, 1, 0, 0]
    max = [
        [0, 0, 1, 2],
        [1, 7, 5, 0],
        [2, 3, 5, 6],
        [0, 6, 5, 2],
        [0, 6, 5, 6]
    ]
    allocation = [
        [0, 0, 1, 2],
        [1, 4, 2, 0],
        [1, 3, 5, 4],
        [0, 6, 3, 2],
        [0, 0, 1, 4]
    ]

    safe_seq = bankers_algo(process, available, max, allocation)
    if safe_seq:
        print("\nThe SAFE Sequence is")
        print(" -> ".join(safe_seq))
    else:
        print("\nSystem is in UNSAFE state!")

main()
```

Output:

```
================================
Need Matrix
P0          0    0    0    0
P1          0    3    3    0
P2          1    0    0    2
P3          0    0    2    0
P4          0    6    4    2

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

Result:

Hence the safe sequence using Banker's algorithm for deadlock Avoidance has been found successfully.