Ex. No.: 9 **231901039**

Date: 2/4/2025

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:

```
#include <stdio.h>
#define MAX 10

int main() {
   int n, m, i, j, k;
   int alloc[MAX][MAX], max[MAX][MAX],
   need[MAX][MAX], avail[MAX];
   int finish[MAX] = {0}, safeSeq[MAX];
   int count = 0;

   printf("Enter the number of processes:");
   scanf("%d", &n);

   printf("Enter the number of resource types: ");
   scanf("%d", &m);

   printf("\nEnter the allocation matrix:\n");
   for (i = 0; i < n; i++) {
       printf("Process %d: ", i);
       for (j = 0; j < m; j++) {
           scanf("%d", &alloc[i][j]);
       }
   }

   printf("\nEnter the maximum matrix:\n");
   for (i = 0; i < n; i++) {
```

```c
        printf("Process %d: ", i);
        for (j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    printf("\nEnter the available resources:\n");
    for (i = 0; i < m; i++) {
        scanf("%d", &avail[i]);
    }

    // Calculate the need matrix
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            need[i][j] = max[i][j] - alloc[i][j];
        }
    }

    // Banker's algorithm
    while (count < n) {
        int found = 0;
        for (i = 0; i < n; i++) {
            if (!finish[i]) {
                int canAllocate = 1;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]) {
                        canAllocate = 0;
                        break;
                    }
                }

                if (canAllocate) {
                    for (k = 0; k < m; k++) {
                        avail[k] += alloc[i][k];
                    }
                    safeSeq[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }

        if (!found) {
            printf("\nSystem is not in a safe state.\n");
            return 0;
        }
    }

    printf("\nSystem is in a safe state.\nSafe sequence is: ");
    for (i = 0; i < n; i++) {
        printf("P%d ", safeSeq[i]);
    }
    printf("\n");

    return 0;
}
```

Sample Output:

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

Result:

Using Banker’s algorithm safe sequence has been found for deadlock avoidance