Ex. No.: 9 **DEADLOCK AVOIDANCE**

Date: ***01.04.2025***

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 processes 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. Update: work = work + allocation[i].
5. Set finish[i] = true and go to step 2.
6. If finish[i] == true for all i, then a safe sequence exists. Print the safe sequence.
7. Else, print that no safe sequence exists (i.e., deadlock may occur).

Program Code (bankers.c):

```
#include <stdio.h>

#define P 5

#define R 3


int main() {

   int allocation[P][R] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};

   int max[P][R] = {{7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}};

   int available[R] = {3, 3, 2};


   int need[P][R], finish[P] = {0}, safeSeq[P];
```

```
int work[R];

// Calculate Need matrix
for (int i = 0; i < P; i++)
   for (int j = 0; j < R; j++)
      need[i][j] = max[i][j] - allocation[i][j];

// Initialize work as available
for (int i = 0; i < R; i++)
   work[i] = available[i];

int count = 0;
while (count < P) {
   int found = 0;
   for (int i = 0; i < P; i++) {
      if (!finish[i]) {
         int j;
         for (j = 0; j < R; j++)
            if (need[i][j] > work[j])
               break;
         if (j == R) {
            for (int k = 0; k < R; k++)
               work[k] += allocation[i][k];
            safeSeq[count++] = i;
            finish[i] = 1;
            found = 1;
         }
      }
   }
```

```
        if (!found) {
            printf("System is not in a safe state.\n");
            return 1;
        }
    }

    printf("The SAFE Sequence is:\n");
    for (int i = 0; i < P; i++)
        printf("P%d ", safeSeq[i]);
    printf("\n");

    return 0;
}
```

Sample Output:

The SAFE Sequence is:

P1 P3 P4 P0 P2

Result:

Thus, the Banker's Algorithm was successfully implemented to determine the safe sequence for deadlock avoidance.