Ex. No.: 9
Date: 3/4/25

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<= 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>
#include <stdbool.h>
#define PRO 5
#define RES 3
int available[RES] = {3,3,2};
int max[PROC][RES] = { {7,5,3}, {3,2,2}, {9,0,2}, {2,2,2},
                       {4,3,3}};
int allocation[PROC][RES] = { {0,1,0}, {2,0,0}, {3,0,2}, {2,1,1},
                              {0,0,2}};
int need[PROC][RES];
bool finish[proc] = {false};
int safe_seq[proc];

void calculate_need(){
    for(int i=0; i<PRO; i++){
        for(int j=0; j<RES; j++){
            need[i][j] = max[i][j] - allocation[i][j];
}   }   }
```

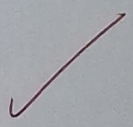
```
bool is_safe(){
    int work[RES];
    for(int i=0; i<RES; i++){
        work[i] = available[i];
    }
    int count=0;
    while(count<PRO){
        bool found = false;
        for(int i=0; i<PRO; i++){
            if(!finish[i]){
                bool can_allocate = true;
                for(int j=0; j<RES; j++){
                    if(need[i][j] > work[j]){
                        can_allocate = false;
                        break;
                    }
                }
                if(can_allocate){
                    for(int j=0; j<RES; j++)
                        work[j] += allocation[j];
                    }
                    safe_seq[count++] = i;
                    finish[i] = true;
                    found = true;
                }
            }
        }
        if(!found)
            break;
    }
    return (count == pro);
}
```

```
int main() {
    calculate_need();

    if (is_safe()) {
        printf("Safe sequence: ");
        for (int i = 0; i < PRO; i++) {
            printf("P%d ", safe_seq[i]);
        }
        printf("\n");
    }
    else {
        printf("No safe sequence.\n");
    }
    return 0;
}
```

OUTPUT:

Safe sequence: P1 P3 P4 P0 P2



Sample Output:

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

Result: A C code is implemented to find if there is a safe sequence using Banker's Algorithm for deadlock avoidance.