Ex. No.: 9
Date: 5/4/25

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_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>
#include<stdbool.h>
int main(){
        int max[5][4];
        int alloc[5][4];
        int need[5][4];
        int available[4];
        printf("\ Available ---");
        for(int i=0 ; i<4 ; i++){
              printf("Enter array element.");
              scanf("%d", &available[i]);
        }
        for(int i=0;i<5; i++){
              for(int j=0 ;j<4; j++){
                    printf(",Enter element");
                    scanf("%d", &max[i][j]);
              }
        }        }
```

```
for (int i=0; i<5; i++){
        for(int j=0; j<4; j++){
                printf("Enter element");
                scanf("%d", & alloc [i][j]);
        }
}
for (int i=0; i<5; i++){
        for (int j=0; j<4; j++){

                need [i][j] = max[i][j] - alloc[i][j];
                printf("%d", need[i][j]);
        }
        printf("\n");
}
int work [4];
bool finish[5];
for (int i=0; i<4 ; i++){
        work [i] = available [i];
}
for (int i=0; i<5 ; i++){
        finish [i] = false;
}
int safeseq [5];
int c=0;
While (c < 5){
        bool found = false;
        for (int i =0; i<5; i++){
                if (finish[i]== false){
                        int j;
                        for (j=0; j<4; j++){
                        if (need[i][j]>work[j])
                                break;
                        }
                        if (j==4){
                                for (int k=0, k<4 ; k++){
                                        work[k]+=alloc[i][j]
                                }
```

```
                    safeseq [c++] = i;
                    finish [i] = true;
                    found = true;
               }
          }
     }
     if (!found){
          printf ("\n System is not safe");
          return;
     }
}
printf ("\n safe sequence: ");
for (int i=0; i<5; i++){
          printf ("P%d", safeseq[i]);
     if (i!=4)
          printf ("→ ");
}
}
```

Sample Output:

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

Available---
1520

	Max				Allocation				Need			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	0	0	0	0
P_1	1	7	5	0	1	0	0	0	0	7	5	0
P_2	2	3	5	6	1	3	5	4	1	0	0	2
P_3	0	6	5	2	0	6	3	2	0	0	2	0
P_4	0	6	5	6	0	0	1	4	0	6	4	2

O.P:- Safe Sequence :
$P_0 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1$

Result:

Hence the safe sequence using banker's algorithm has been generated for deadlock avoidance successfully.