Ex. No.: 9
Date: 03-04-25

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 n,m;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter number of resources: ");
    scanf("%d", &m);
    int allocation[n][m], max[n][m], need[n][m], available[m];
    printf("Enter allocation matrix:\n");
    for (int i=0; i<n; i++) {
        for int j=0; j<m; j++) { scanf("%d", &allocation[i][j]
        }
    }

    printf("Enter Max Matrix:\n");
```

```
for (int i=0; i<n; i++){
        for (int j=0; j<m; j++){
            scanf("%d", &max[i][j]);
        }
}

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

for (int i=0; i<n; i++){
    for (int j=0; j<m; j++){
        need[i][j] = max[i][j] - allocation[i][j];
    }
}

bool finish[n];
for (int i=0; i<n; i++) {
        finish[i] = false;
}

int safeSequence[n], count=0;
int work[m];
for (int i=0; i<m; i++){
        work[i] = available[i];
}
```

```
while ( count <n){

    bool found = false;
    for (int i=0; i<n ; i++){
            if ( ! finish[i]){
                    bool canRun = true;
                    for (int j=0; j<m; j++){
                          if ( need [i][j] > work [j]){
                                   canRun = false;
                                   break;
                          }
                    }
                    if (canRun){
                        for (int j=0; j<m; j++) {
                               work [j] += allocation[i][j];
                        }
                        safeSequence[count ++] == i;
                        finish [i] = true;
                        found = true;
                    }
            }
    }
    if(! found){
            printf( "system is NOT in a safe state. No
                    safe sequence.\n");
            return 0;
    }
}
printf ("system is in a safee state. \n safe sequence: ");

for (int i=0; i<n; i++) {
        printf("p%d -> ", safeSequence [i]);
}
  printf("\n")
```



```
  return 0;
}
```

Input :

Enter number of processes: 5

Enter number of resources: 3

Enter allocation Matrix :

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

Enter max matrix :

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter available resources

3 3 2

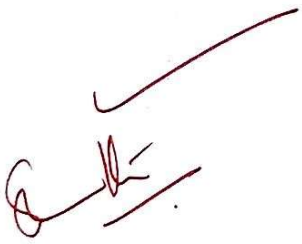
output

System is in a safe state.

Safe sequence : P1 → P3 → P4 → P0 → P2

Sample Output:

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



Result:

Thus, safe sequence using Banker's algorithm for deadlock avoidance executed successfully.