LAB-6

Q) Write a program to Simulate bankers algorithm for Dead Lock Avoidance (Banker‘s Algorithm).

PROGRAM:

```
#include <stdio.h>

#include <stdbool.h>


void main() {

   int alloc[10][10], max[10][10], avail[10], work[10];

   int need[10][10];

   char finish[10] = {0};

   int n, m;

   char safe_sequence[10][3];

   int count = 0;


   printf("Enter the number of processes and resources: ");

   scanf("%d%d", &n, &m);


   printf("Enter the allocation matrix:\n");

   for (int i = 0; i < n; i++)

      for (int j = 0; j < m; j++)

         scanf("%d", &alloc[i][j]);


   printf("Enter the maximum resource matrix:\n");
```

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

printf("Enter the available resource vector: ");
for (int i = 0; i < m; i++) {
    scanf("%d", &avail[i]);
    work[i] = avail[i];
}

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

// Safety Algorithm
bool found = false;
int index = 0;

while (count < n) {
    found = false;
    for (int i = 0; i < n; i++) {
        if (!finish[i]) {
            bool can_execute = true;
            for (int j = 0; j < m; j++) {
                if (need[i][j] > work[j]) {
```

```
                can_execute = false;
                break;
            }
        }
        if (can_execute) {
            for (int j = 0; j < m; j++)
                work[j] += alloc[i][j];
            finish[i] = 1;
            sprintf(safe_sequence[index++], "P%d", i + 1);
            count++;
            found = true;
        }
    }
  }
  if (!found)
    break;
}

if (count == n) {
  printf("System is in a safe state.\nSafe sequence: ");
  for (int i = 0; i < n; i++) {
    printf("%s", safe_sequence[i]);
    if (i < n - 1)
      printf(" -> ");
  }
  printf("\n");
```

```
    } else {
        printf("System is not in a safe state.\n");
    }
}
```

OUTPUT: