Name: Sahin Nayak **Enrollment Number : 12023006015086**

ASSIGNMENT-6

1. For deadlock avoidance, write a C program to simulate the Bankers algorithm. DESCRIPTION: In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, other waiting processes hold the resources a waiting process has requested, preventing it from changing state again. We refer to this situation as a deadlock. Deadlock avoidance is one of the techniques for handling deadlocks. This approach necessitates providing the operating system with additional resources beforehand, as well as information about which resources a process will request and use during its lifetime. With this additional knowledge, it can decide for each request whether or not the process should wait. The system considers the resources currently available, the resources allocated to each process, and the future requests and releases of each process to determine whether to satisfy the current request or delay it. Banker's algorithm is a deadlock avoidance algorithm that is applicable to a system with multiple instances of each resource type.

```
#include <stdio.h>
#define MAX_PROCESSES 10
#define MAX_RESOURCES 10
int isSafe(int processes[], int available[], int max[][MAX_RESOURCES],
   int allocation[][MAX_RESOURCES], int need[][MAX_RESOURCES], int n, int m) {
   int work[MAX_RESOURCES], finish[MAX_PROCESSES], safeSeq[MAX_PROCESSES];
   int count = 0;
   for (int i = 0; i < m; i++)
      work[i] = available[i];
   for (int i = 0; i < n; i++)
      finish[i] = 0;
   while (count < n) {
      int found = 0;
      for (int p = 0; p < n; p++) {
         if (finish[p] == 0) {
            int canExecute = 1;
            for (int r = 0; r < m; r++) {
               if (need[p][r] > work[r]) {
                  canExecute = 0;
                  break; }}
            if (canExecute) {
               for (int r = 0; r < m; r++)
                  work[r] += allocation[p][r];
               safeSeq[count++] = p;
               finish[p] = 1;
               found = 1; }}}
      if (found == 0) {
         printf("System is not in a safe state.\n");
         return 0;
   printf("System is in a safe state.\nSafe sequence is: ");
   for (int i = 0; i < n; i++)
      printf("%d ", safeSeq[i]);
   printf("\n");
   return 1;}
int main() {
   int n, m; // Number of processes and resources
   int processes[MAX_PROCESSES], available[MAX_RESOURCES];
   int max[MAX_PROCESSES][MAX_RESOURCES], allocation[MAX_PROCESSES][MAX_RESOURCES];
   int need[MAX_PROCESSES][MAX_RESOURCES];
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   printf("Enter the number of resource types: ");
   scanf("%d", &m);
   for (int i = 0; i < n; i++) {
      processes[i] = i;}
   printf("Enter the available instances of each resource:\n");
   for (int i = 0; i < m; i++) {
      scanf("%d", &available[i]);}
   printf("Enter the maximum resource demand for each process:\n");
   for (int i = 0; i < n; i++) {
      printf("For process %d:\n", i);
      for (int j = 0; j < m; j++) {
         scanf("%d", &max[i][j]); }}
   printf("Enter the allocated resources for each process:\n");
   for (int i = 0; i < n; i++) {
      printf("For process %d:\n", i);
      for (int j = 0; j < m; j++) {
```

Name: Sahin Nayak **Enrollment Number : 12023006015086**

```
        scanf("%d", &allocation[i][j]); }}
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            need[i][j] = max[i][j] - allocation[i][j]; }}
    isSafe(processes, available, max, allocation, need, n, m);
    return 0; }
```

```
sahin@sahin-VirtualBox:~$ gedit ass6.c
sahin@sahin-VirtualBox:~$ gcc ass6.c -o ass6
sahin@sahin-VirtualBox:~$ ./ass6
Enter the number of processes: 5
Enter the number of resource types: 3
Enter the available instances of each resource:
3 3 2
Enter the maximum resource demand for each process:
For process 0:
7 5 3
For process 1:
3 2 2
For process 2:
9 0 2
For process 3:
2 2 2
For process 4:
4 3 3
Enter the allocated resources for each process:
For process 0:
0 1 0
For process 1:
2 0 0
For process 2:
3 0 2
For process 3:
2 1 1
For process 4:
0 0 2
System is in a safe state.
Safe sequence is: 1 3 4 0 2
sahin@sahin-VirtualBox:~$
```