Assignment - 06
TOPIC: Deadlock Avoidance

1.For deadlock avoidance, write a C program to simulate the Bankers algorithm.

->Ans:

```
#include <stdio.h>
#define MAX_PROCESSES 5
#define MAX_RESOURCES 3
int allocate[MAX_PROCESSES][MAX_RESOURCES]; // Allocated resources matrix
int max[MAX_PROCESSES][MAX_RESOURCES];      // Maximum demand matrix
int need[MAX_PROCESSES][MAX_RESOURCES];     // Remaining needs matrix
int available[MAX_RESOURCES];               // Available resources vector
int safeSequence[MAX_PROCESSES];            // Safe sequence of processes
// Function to check if the current system state is safe
int isSafe(int processes[], int nProcesses, int resources) {
    int finish[MAX_PROCESSES] = {0}; // Keep track of finished processes
    int work[MAX_RESOURCES];         // Temporary work array to track available
resources
    for (int i = 0; i < resources; i++)
        work[i] = available[i];      // Initialize work with the available
resources
    int count = 0;
    while (count < nProcesses) {
        int found = 0;
        for (int i = 0; i < nProcesses; i++) {
            if (finish[i] == 0) { // Process not finished yet
                int j;
                for (j = 0; j < resources; j++) {
                    if (need[i][j] > work[j])
                        break;
                }
                if (j == resources) { // If all needs are less than available work
                    for (int k = 0; k < resources; k++)
                        work[k] += allocate[i][k]; // Release resources

                    safeSequence[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }
        if (found == 0) {
            printf("System is not in a safe state.\n");
            return 0;
        }
    }
    return 1; // Safe state
}

int main() {
```

```
    int processes[MAX_PROCESSES] = {0, 1, 2, 3, 4}; // Process IDs
    int nProcesses = 5, nResources = 3;

    // Example input
    printf("Enter allocation matrix:\n");
    for (int i = 0; i < nProcesses; i++) {
        for (int j = 0; j < nResources; j++) {
            scanf("%d", &allocate[i][j]);
        }
    }
    printf("Enter maximum matrix:\n");
    for (int i = 0; i < nProcesses; i++) {
        for (int j = 0; j < nResources; j++) {
            scanf("%d", &max[i][j]);
        }
    }
    printf("Enter available resources:\n");
    for (int i = 0; i < nResources; i++) {
        scanf("%d", &available[i]);
    }
    // Calculate the need matrix
    for (int i = 0; i < nProcesses; i++) {
        for (int j = 0; j < nResources; j++) {
            need[i][j] = max[i][j] - allocate[i][j];
        }
    }
    // Check system safety
    if (isSafe(processes, nProcesses, nResources)) {
        printf("System is in a safe state.\nSafe sequence is: ");
        for (int i = 0; i < nProcesses; i++) {
            printf("%d ", safeSequence[i]);
        }
        printf("\n");
    } else {
        printf("System is not in a safe state.\n");
    }
    return 0;
}
```

Output:

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ vim bankersAlgorithm.c
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ gcc -o bankersAlgorithm bankersAlgorithm.c
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ./bankersAlgorithm
Enter allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter maximum matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter available resources:
3 3 2
System is in a safe state.
Safe sequence is: 1 3 4 0 2
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$
```