NAME: TEJAS NALAWADE PRACTICAL NO: 6 ROLL NO: TCOD01

Assignment Title: Perform Implementation of Deadlock avoidance algorithm, i.e Bankers Algorithms.
Problem Statement: Write a Java program to implement Banker's Algorithm.

CODE:

```
// Bankers Algorithm
import java.util.Scanner;

public class BankersAlgorithm {
    public static int[] IsSafeSequence(int[][] allocation, int[][] need, int[] available) {
        int n = allocation.length; // Number of processes
        int m = available.length; // Number of resources
        boolean[] finished = new boolean[n];
        int[] newAvailable = available.clone();
        int[] safeSequence = new int[n];
        int count = 0;
        boolean found;
        while (count < n) {
            found = false;
            for (int i = 0; i < n; i++) {
                if (!finished[i]) {
                    boolean canAllocate = true;
                    for (int j = 0; j < m; j++) {
                        if (need[i][j] > newAvailable[j]) {
                            canAllocate = false;
                            break;
                        }
                    }
                    if (canAllocate) {
                        for (int j = 0; j < m; j++) {
                            newAvailable[j] += allocation[i][j];
                        }
                        safeSequence[count++] = i;
                        finished[i] = true;
                        found = true;
                    }
                }
            }
            if (!found) {
                return null;
            }
        }
        return safeSequence;
    }


  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter number of processes: ");
    int numProcesses = scanner.nextInt();
    System.out.print("Enter number of resources: ");
```

```
        int numResources = scanner.nextInt();

        // Array Dewcalreation
        int[][] allocation = new int[numProcesses][numResources];
        int[][] max = new int[numProcesses][numResources];
        int[][] need = new int[numProcesses][numResources];
        int[] available = new int[numResources];

        // Allocation Matrix input
        System.out.println("Enter allocation matrix:");
        for (int i = 0; i < numProcesses; i++) {
            for (int j = 0; j < numResources; j++) {
                allocation[i][j] = scanner.nextInt();
            }
        }

        // Max Matrix input
        System.out.println("Enter max matrix:");
        for (int i = 0; i < numProcesses; i++) {
            for (int j = 0; j < numResources; j++) {
                max[i][j] = scanner.nextInt();
            }
        }

        // (Need Matrix) Substraction of Matrix
        for (int i = 0; i < numProcesses; i++) {
            for (int j = 0; j < numResources; j++) {
                need[i][j] = max[i][j] - allocation[i][j];
            }
        }

        System.out.print("Enter available resources: ");
        for (int i = 0; i < numResources; i++) {
            available[i] = scanner.nextInt();
        }

        // Safe  Sequence
        int[] safeSequence = IsSafeSequence(allocation, need, available);
        if (safeSequence != null) {
            System.out.print("Safe sequence is: ");
            for (int i = 0; i < safeSequence.length; i++) {
                    System.out.print("P" + (safeSequence[i] + 1));
                if (i != safeSequence.length - 1) System.out.print(" -> ");
            }
            System.out.println();
        } else {
            System.out.println("No safe sequence exists. System is not in a safe state.");
        }
    }
}
```

OUTPUT:

```
PRACTICAL\CODE\Bankers-Algorithm on ᛘ main [?] via ☕ v24.0.2
❯ java BankersAlgorithm.java
Enter number of processes: 5
Enter number of resources: 4
Enter allocation matrix:
0 0 1 2
2 0 0 0
0 0 3 4
2 3 5 4
0 3 3 2
Enter max matrix:
0 0 1 2
2 7 5 0
6 6 5 6
4 3 5 6
0 6 5 2
Enter available resources: 2 1 0 0
Safe sequence is: P1 → P4 → P5 → P2 → P3

PRACTICAL\CODE\Bankers-Algorithm on ᛘ main [?] via ☕ v24.0.2 took 1m18s
❯ java BankersAlgorithm.java
Enter number of processes: 5
Enter number of resources: 4
Enter allocation matrix:
0 0 1 2
2 0 0 0
0 1 3 4
2 3 5 4
0 3 3 2
Enter max matrix:
0 0 1 2
2 7 5 0
6 6 5 6
4 3 5 6
0 6 5 2
Enter available resources: 2 0 0 0
No safe sequence exists. System is not in a safe state.
```