**Exp. 11**

**Problem Statement:**

Write a program to implement Banker’s algorithm for deadlock avoidance.

Code: -

import java.util.Scanner;

public class BankersAlgorithm {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input number of processes and resources

System.out.print("Enter the number of processes: ");

int n = sc.nextInt();

System.out.print("Enter the number of resources: ");

int m = sc.nextInt();

// Input available resources

int[] available = new int[m];

System.out.println("Enter the available resources:");

for (int i = 0; i < m; i++) {

available[i] = sc.nextInt();

}

// Input allocation matrix

int[][] allocation = new int[n][m];

System.out.println("Enter the allocation matrix:");

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

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

allocation[i][j] = sc.nextInt();

}

}

// Input maximum matrix

int[][] maximum = new int[n][m];

System.out.println("Enter the maximum matrix:");

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

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

maximum[i][j] = sc.nextInt();

}

}

// Calculate the need matrix

int[][] need = new int[n][m];

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

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

need[i][j] = maximum[i][j] - allocation[i][j];

}

}

// Safety algorithm

boolean[] finish = new boolean[n];

int[] safeSequence = new int[n];

int index = 0;

while (true) {

boolean found = false;

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

if (!finish[i]) {

boolean safe = true;

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

if (need[i][j] > available[j]) {

safe = false;

break;

}

}

if (safe) {

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

available[j] += allocation[i][j];

}

finish[i] = true;

safeSequence[index++] = i;

found = true;

}

}

}

if (!found) {

break;

}

}

// Check if all processes could finish

boolean safe = true;

for (boolean f : finish) {

if (!f) {

safe = false;

break;

}

}

if (safe) {

System.out.println("System is in a safe state.");

System.out.print("Safe sequence: ");

for (int i : safeSequence) {

System.out.print("P" + i + " ");

}

} else {

System.out.println("System is NOT in a safe state. Deadlock may occur.");

}

sc.close();

}

}

OUTPUT: -

