## **Practical No. 8: Banker's Algorithm for Deadlock Avoidance**

### **Aim:**

To write a C program that implements **Banker's Algorithm** to check whether the system is in a safe state or not.

### **Key Concepts:**

* **Banker’s Algorithm:** Resource allocation and deadlock avoidance algorithm that checks for system safety.
* **Need Matrix:** Need = Max - Allocation
* **Safe State:** If a safe sequence exists such that all processes can finish, the system is in a safe state.

### **Sample Execution:**

yaml

CopyEdit

Enter number of processes: 4

Enter number of resources: 1

Enter available resources:

Resource 0: 2

Enter allocation matrix:

Process 0: Resource 0: 3

Process 1: Resource 0: 0

Process 2: Resource 0: 2

Process 3: Resource 0: 1

Enter maximum matrix:

Process 0: Resource 0: 8

Process 1: Resource 0: 5

Process 2: Resource 0: 9

Process 3: Resource 0: 5

Safe sequence: P3 P2 P0 P1

The system is in a safe state.

### **Explanation:**

* The algorithm calculates the **Need matrix** by subtracting the Allocation from the Max matrix.
* It checks if resources can be allocated to each process in some sequence such that all processes finish.
* If such a sequence exists, the system is **safe**.

### 

### 

### **Conclusion:**

The Banker's Algorithm was successfully implemented to determine the safe state of a system. It ensures that deadlock is avoided by granting resource allocation only if the system remains in a safe state.