**EXPERIMENT 17**

**Illustrate the deadlock avoidance concept by simulating Banker’s algorithm using C.**

## AIM :

To illustrate the deadlock avoidance concept by simulating Banker’s algorithm using C.

## ALGORITHM :

1. Define Data Structures: Define appropriate data structures to store the available resources, maximum resources, allocated resources, and need matrix for each process.
2. Initialize Matrices: Initialize the available, maximum, allocated, and need matrices based on the system's resources and the maximum demand of each process.
3. Input Request: Implement a function to input resource request from processes. This function should validate if the request is within the maximum limit specified by each process.
4. Safety Algorithm: Implement the Banker's safety algorithm to check if the system is in a safe state. Use the available, allocated, and need matrices to determine if the system can allocate resources to processes without entering into a deadlock state.
5. Resource Allocation: Implement resource allocation functions to handle the request from processes. Check if the request can be granted safely using the Banker's algorithm. If the request can be granted, update the allocated and available matrices accordingly.
6. User Interface: Create a user interface to interact with the program. Allow users to input resource requests and display the current state of the system, including available resources and resource allocation status.
7. Deadlock Scenario: Introduce scenarios where a deadlock can potentially occur (e.g., requesting more resources than available or requesting resources in a circular wait condition).
8. Testing: Test the program with different resource request scenarios. Ensure that the system handles requests properly without entering into a deadlock state. Also, test scenarios where the system should deny requests to prevent deadlock.
9. Documentation (Optional): Add comments and documentation to the code to explain the functionality of different sections. Document the Banker's algorithm steps and how it prevents deadlocks in the system.
10. Error Handling: Implement error handling mechanisms to deal with invalid input, unexpected scenarios, or any issues that might occur during resource allocation and deallocation.

OUTPUT :

