DEADLOCK

In concurrent computing, a deadlock is a state in which each member of a group is waiting for another member, including itself, to take action, such as sending a message or more commonly releasing a lock. Deadlock is a common problem in multiprocessing systems, parallel computing, and distributed systems, where software and hardware locks are used to arbitrate shared resources and implement process synchronization.

**Requirements of a deadlock:**

1. Mutual exclusion

2. Hold and wait

3. No pre-emption

4. Circular wait

**Handling a deadlock:**

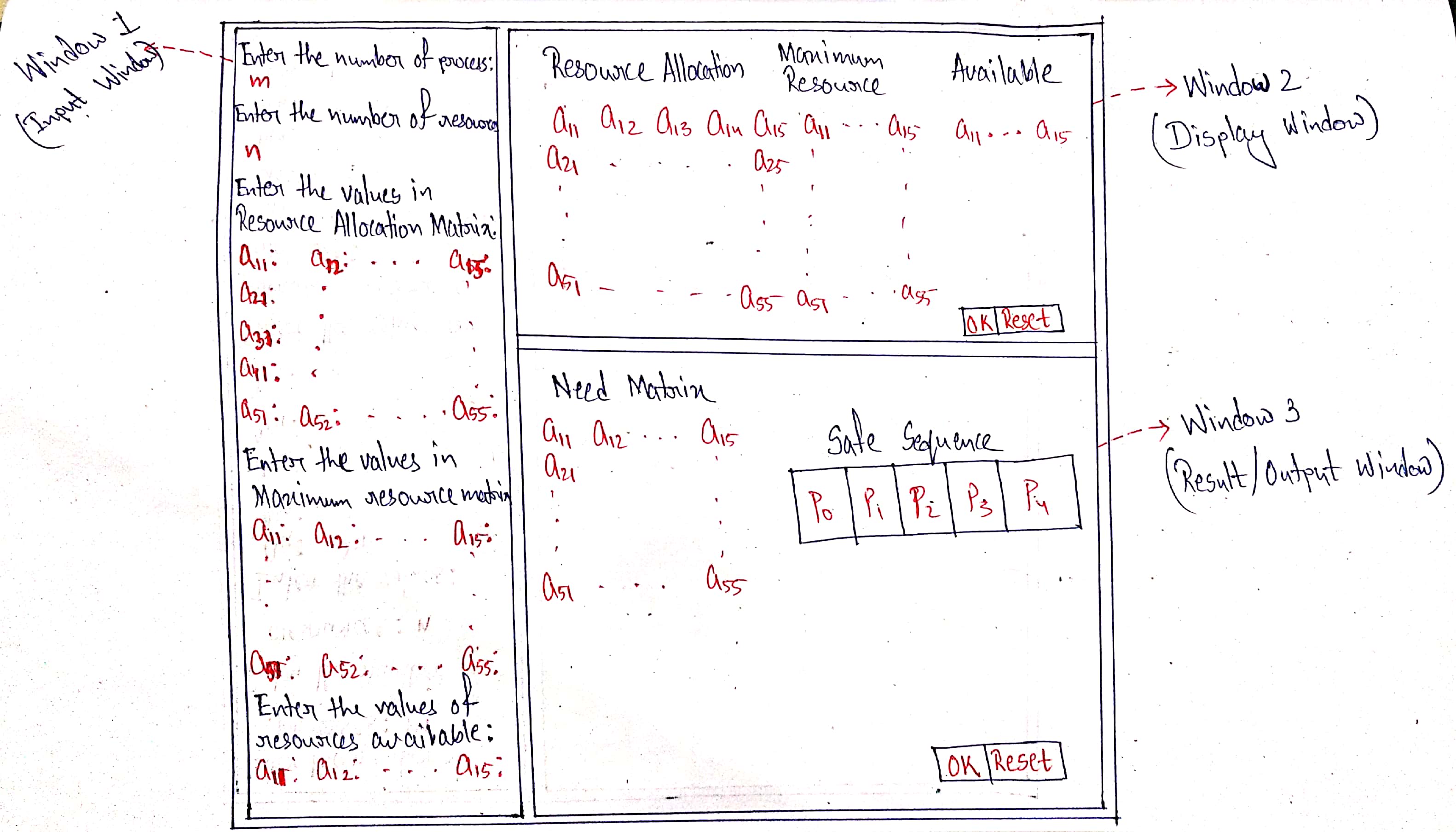
1. Deadlock prevention and avoidance
2. Deadlock detection and recovery
3. Ignore the problem all together

**Banker’s Algorithm:**

It is a resource allocation and deadlock avoidance algorithm that tests for the safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes a ***‘s-state’*** check to test for allocation should be allowed to continue or not.

This algorithm works by taking the number of processes and the number of resources in a matrix. It also needs a matrix which represents the maximum resource and also a matrix of size 1Xn (where 1 = number of row and n = number of columns) which represents the available matrix.

The



The work flow of Banker’s algorithm for demonstration purpose is as follows:

1. Window 1(Input Window):

**Process Input:**

->Here, the user has to enter the number of processes for the demonstration purpose.

->The maximum number of processes allowed for the same is 5, starting from P0,P1…,P4.

->Every process represents the number of rows in the resource allocation matrix as well as the maximum resource matrix.

**Resource Input:**

->Here, the user has to enter the number of resources for the demonstration purpose.

->The maximum number of resources allowed for the same is 5, starting from R1,R2…,R5.

->Every resource represents the number of columns in the resource allocation as well as maximum resource matrix.

**Giving input in the resource allocation matrix:**

->After setting the number of rows and columns for both the matrices (i.e., Resource allocation and maximum resource) ,the resource allocation matrix is created virtually of the size ***m X n*** (as per the user’s input of number of processes and resources) with no values in it.

->The cursor moves automatically to the next element of the matrix after getting input from the user.

**Giving input in the maximum resource matrix:**

->When the resource allocation matrix is filled up by the user’s input, he has to hit the ***‘Enter’*** key to move the cursor to the maximum resource matrix.

->This matrix is same as the previous one, i.e. , it has that virtually created matrix of size ***m X n*** with no values in it, like the resource allocation matrix.

->The cursor moves to the next element automatically to get the input from the user.

->After this matrix is filled too, the user has to hit the ***‘Enter’*** key to move the cursor to the available matrix.

**Giving input in the available matrix:**

->This matrix is created when the user provides the number of resources, he wants in his demonstration, with only a single number of row.

->The cursor moves automatically to the next element when the user gives input.

1. Window 2(Display Window):

In this window the user will be able to view the clear matrix formation on the basis of his inputs in the previous window.

By hitting the ***‘OK’*** button, the user agrees to continue the process of demonstration to carry forward and by hitting the ***‘RESET’*** button the user denies to the same. So the user’s input vanishes from the input window and he has to create the matrix as well as has to enter the values once again.

1. Window 3(Output Window):

->In this window, the ***‘Need Matrix’*** is created basing on the resource allocation and maximum resource matrices as per the following equation:

***Need matrix = Maximum Resource Matrix - Resource Allocation Matrix***

->In the same window, the ***‘Safe Sequence’*** is also showcased one by one in the form of an array.

->When the user hits the ‘OK’ key, the process of demonstration is stopped and the user is taken to the previous page and when the user hits the ***‘RESET’*** key, all the inputs, matrices and the output are erased but the user remains on the same page.