**CSCE-5640**

**Operating System Design**

**Project**

**Deadlock detection algorithm:**

Step 1:

Obtain resource allocation graph from the user

Number of process nodes

Number of resource nodes

Resources allocated (matrix)

Resources requested (matrix)

Resource capacities (vector)

Step 2:

Present inputs obtained to the user to verify if required

Step 3:

Identify if there are any sinks in the given graph and reduce such nodes. Update the process state vector appropriately.

Step 4:

Calculate current resource vector by releasing the resources previously held by the sinks.

Step 5:

Check if there exists a connected node on the remaining graph.

If yes,

Loop through all processes

check if we can allocate the requested resources to the process.

If yes,

Allow the process to finish and release the previously held resource to the current resource vector.

Else

Go back to step 5

End of loop – Graph is not completely reducible, so system Is in a deadlock state. Print the

irreducible processes from the process state vector.

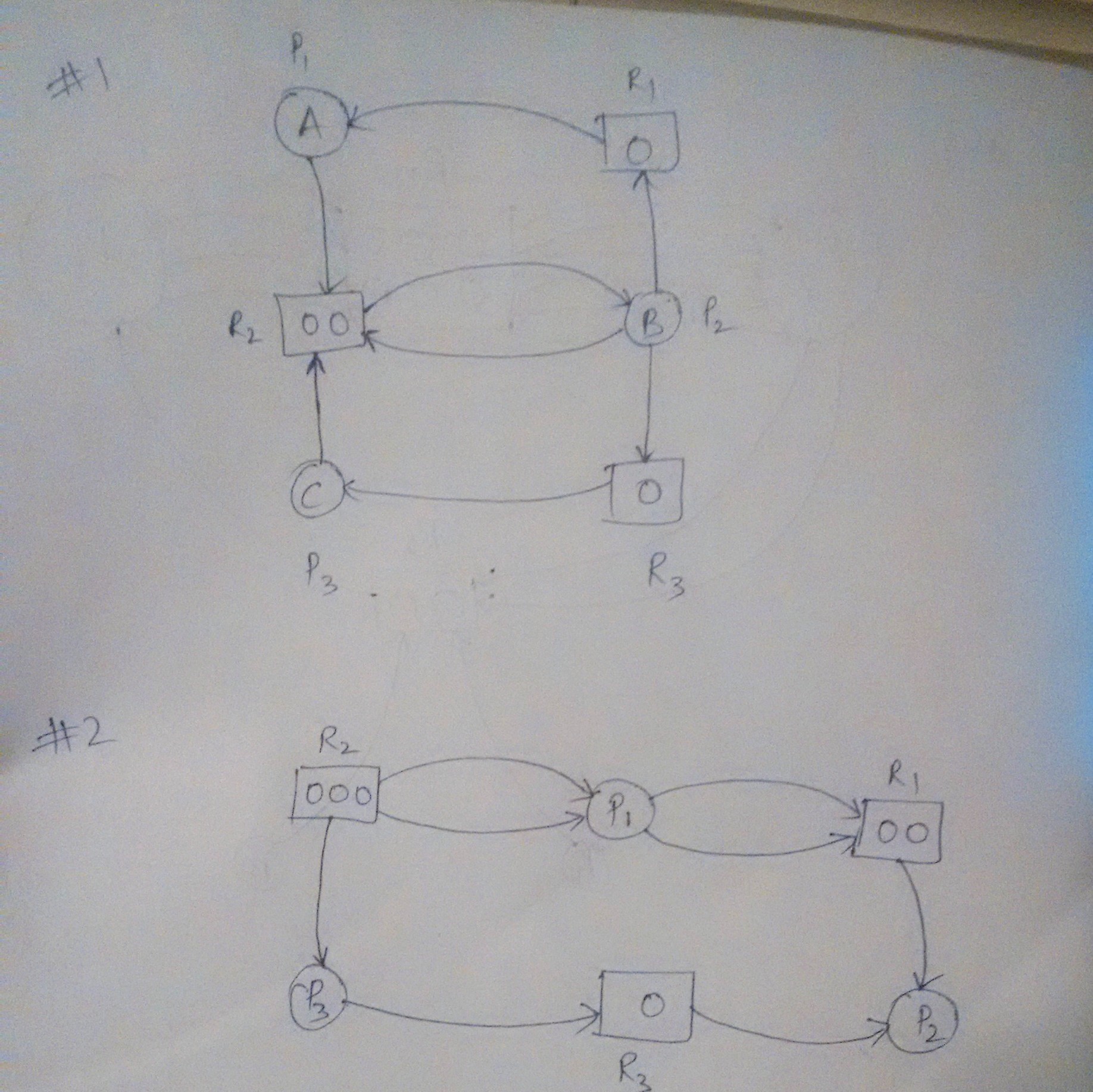
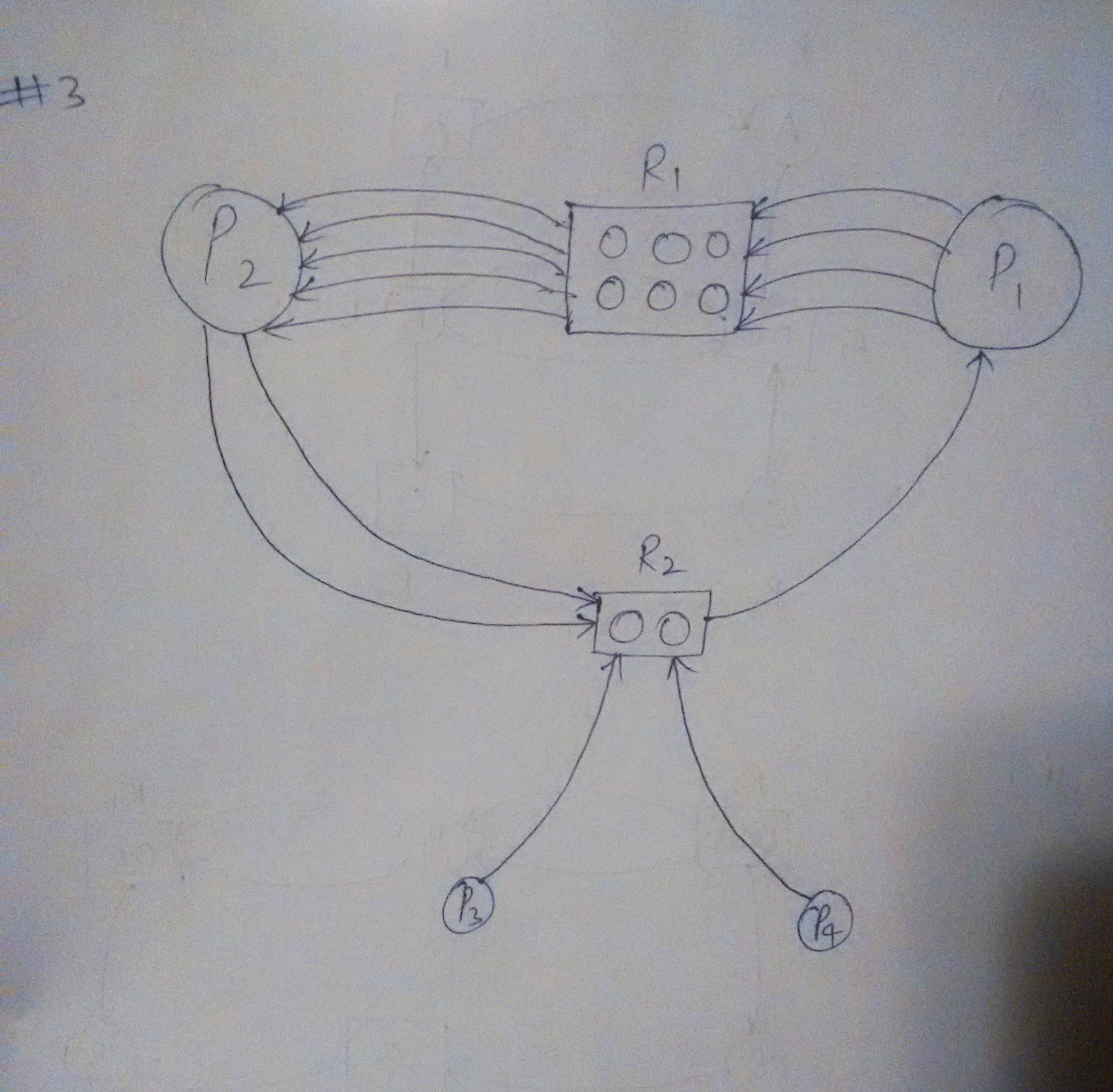
Exit

Else

Graph is completely reduced and Is deadlock free.

Exit

**Example graphs:**



**Example#1**

No of processes: 3

No of resources: 3

Resources allocated matrix: 1 0 0

0 1 0

0 0 1

Resource request matrix: 0 1 0

1 1 1

0 1 0

Resource capacity vector: 1 2 1

Result: No deadlock

**Example#2**

No of processes: 3

No of resources: 3

Resources allocated matrix: 0 2 0

1 0 1

0 1 0

Resource request matrix: 2 0 0

0 0 0

0 0 1

Resource capacity vector: 2 3 1

Result: No Deadlock

**Example#3**

No of processes: 4

No of resources: 2

Resources allocated matrix: 0 1

5 0

0 0

0 0

Resource request matrix: 4 0

0 2

0 1

0 1

Resource capacity vector: 6 2

Result: Deadlock detected

**Potential improvements:**

1. This algorithm assumes that there would not be any consumable resources in the graph, which is not necessarily the case in a real-world situation. Hence, we can improve the algorithm further to include such resources
2. Error checking and validation can be improved to make the solution more readable.
3. Time complexity of the portion where we check for reducible nodes is O(n^2) has been employed. This however is not efficient and better techniques could be adopted making the algorithm better.
4. Input in this algorithm is assumed to be well organized with all the process nodes, resource nodes, resource capacity, allocation state and request states clearly identified and formatted. The algorithm can be improved to receive only a single adjacency matrix with the bare minimum information and the algorithm figures out the rest that is needed to detect potential deadlocks. Even better, a graph represented as an ordered pair of edges.