

# **OPERATING SYSTEMS(A8510)**

## **TOPIC:-DEADLOCKS**

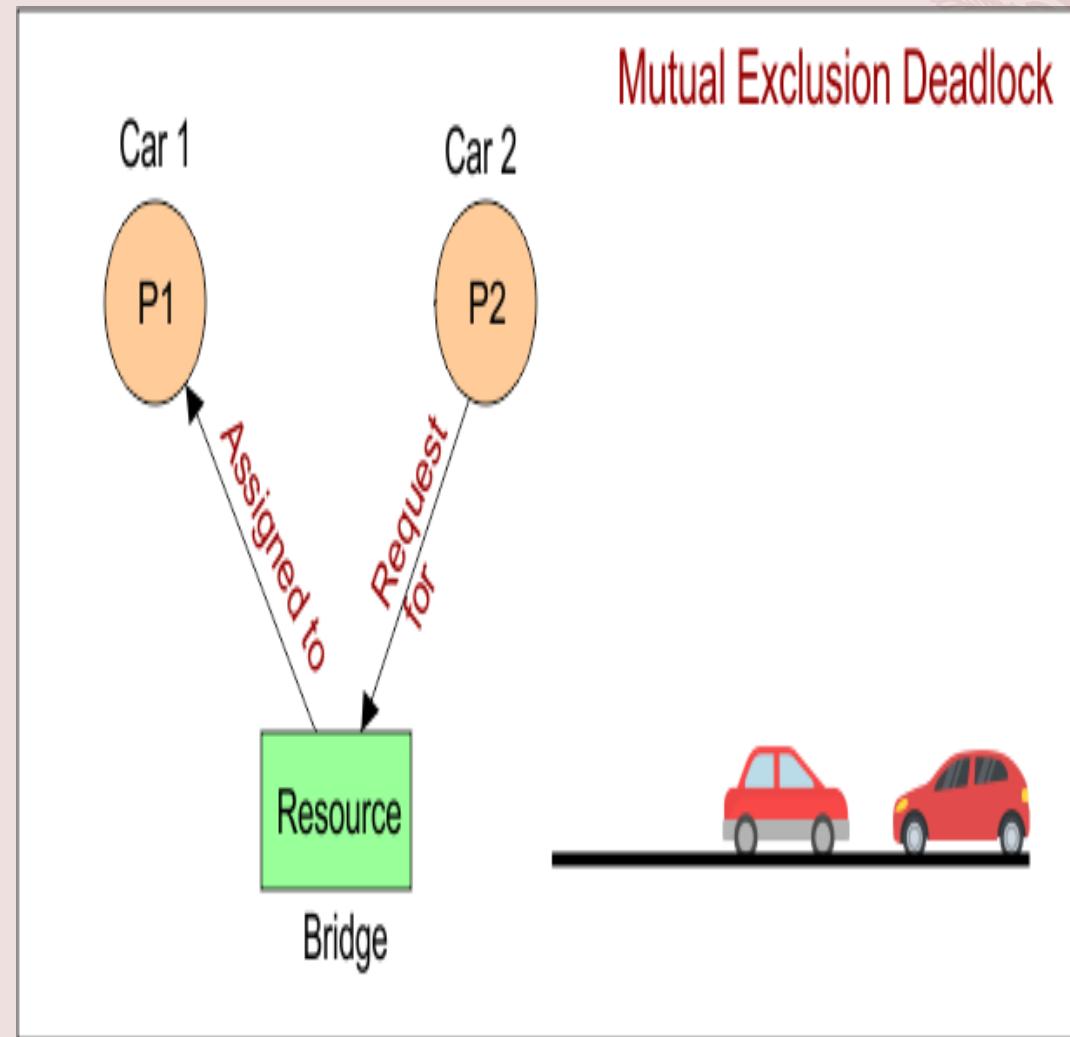
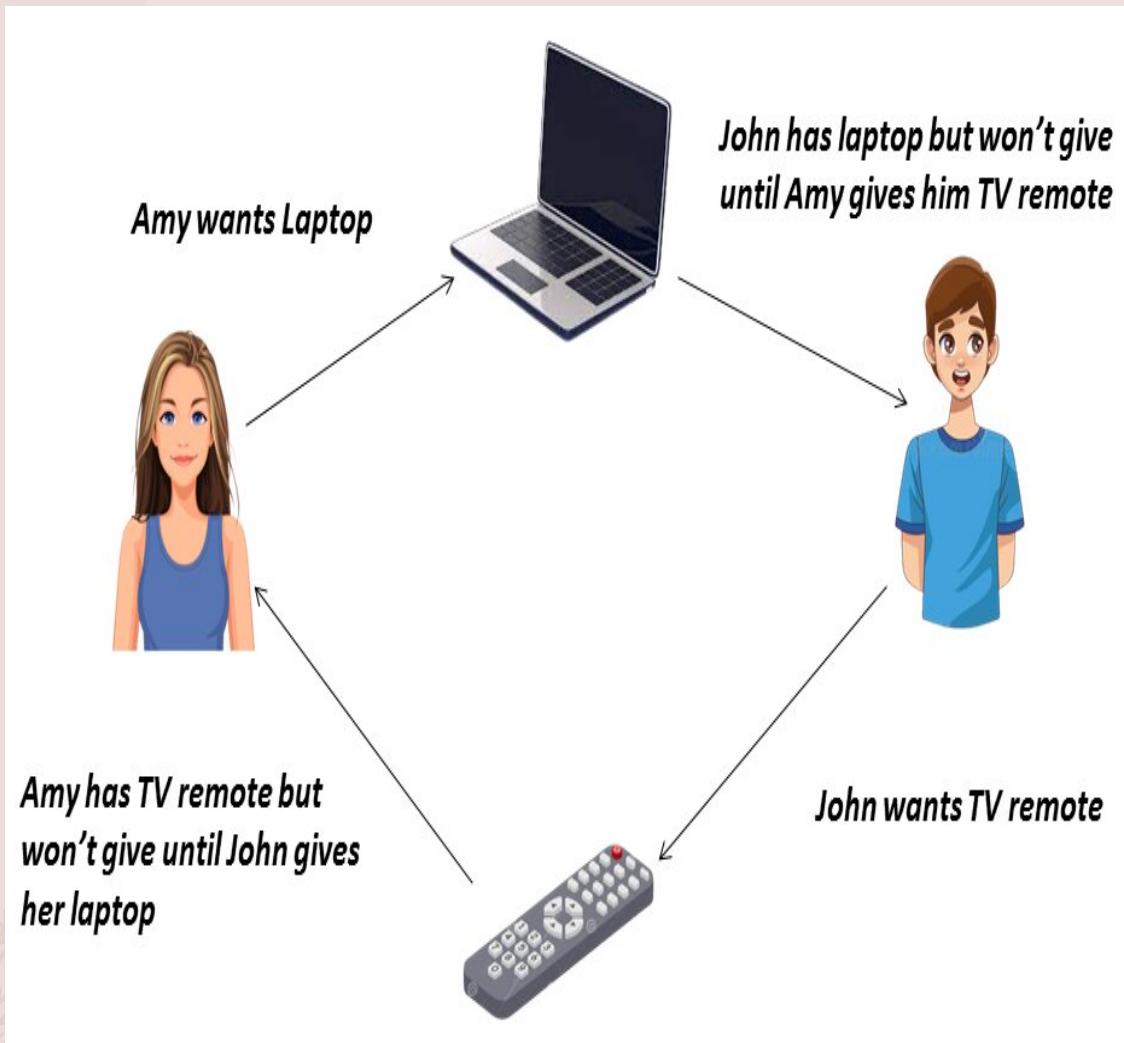
<b>NAME</b>	<b>ROLL NO:</b>
B.CHIRANJEEVI	24881A1276
B.SRI MANI HARIBA	24881A1278
D.ARCHANA	24881A1289

# DEFINATION AND REAL LIFE EXAMPLES ON DEADLOCKS:

- ✓ A state where two or more processes are permanently waiting for resources held by others.
- ✓ It's a mutual dependence that leads to system paralysis.
- ✓ The Traffic Intersection Deadlock.
- ✓ The Kitchen Appliance Deadlock.
- ✓ Office Printer Scenario.



## Mutual Exclusion Deadlock



## THE FOUR PILLARS OF DEADLOCK (CONDITIONS):

- 1) **Mutual Exclusion:** Resources are non-sharable (e.g., a printer). Only one process can use it at a time.
- 2) **Hold and Wait:** A process is holding at least one resource while waiting to acquire additional resources held by other processes.
- 3) **No Preemption:** Resources cannot be forcibly removed from a process; they must be released voluntarily.
- 4) **Circular Wait:** A closed chain of processes exists, where each process in the chain is waiting for a resource held by the next process.

## **VISUALIZING THE PROBLEM:**

**Purpose:** A powerful tool for modeling resource requests and assignments.

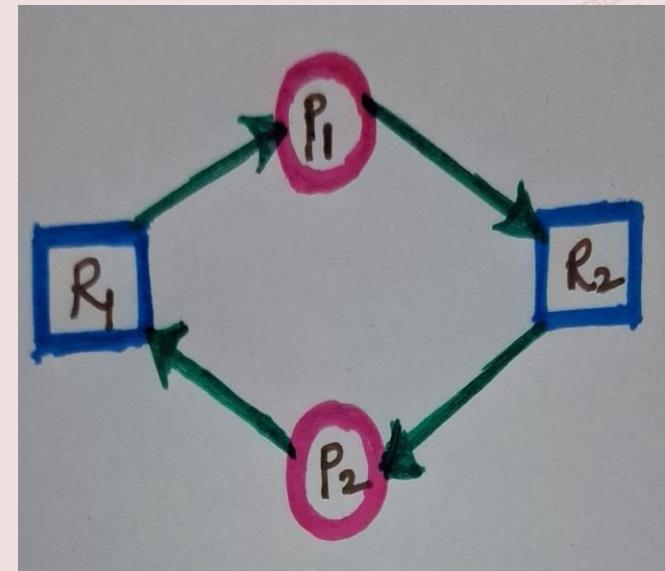
**Components:**

**Circles:** Processes ()

**Squares:** Resources ()

**Request Edge:** (Process wants the resource)

**Assignment Edge:** (Resource is held by the process)



## **THREE STRATEGIES FOR HANDLING DEADLOCKS:**

**Deadlock Prevention (Proactive):** Design the system to structurally violate one of the four necessary conditions (expensive).

**Deadlock Avoidance (Dynamic):** Check every resource request at runtime to ensure the system can remain in a "safe state" (moderate overhead).

**Deadlock Detection & Recovery (Reactive):** Allow deadlocks to occur, periodically detect them, and then forcibly recover the system (lowest runtime overhead, but potential data loss).

## Deadlock Prevention:

### 1) Targeting Hold and Wait:

Option A (Extreme): Force processes to request **all** resources upfront.

Option B (Better): Process must release **all** currently held resources before requesting new ones.

### 2) Targeting Circular Wait (The Most Practical):

Impose a global, hierarchical ordering on all resource types.

Processes can only request resources in **increasing order** of enumeration.

## Deadlock Avoidance (The Banker's Algorithm):

Requires processes to state their maximum resource needs in advance.  
The OS simulates the request: if granting the resource leaves the system in a safe state, the request is approved. Otherwise, it is delayed.

**Benefit:** Highly efficient resource utilization.

**Drawback:** Requires prior knowledge of resource needs, which is often impossible in real-world

## Detection and Recovery:

### Detection:

Periodically check for the existence of a cycle in the **Wait-For Graph**. If a cycle is found, a deadlock exists.

### Recovery(Breaking the Cycle):

- 1) **Process Termination:** Abort one or more deadlocked processes. The cost is the loss of work already done.
- 2) **Resource Preemption:** The victim process must be returned to a previous, safe state to restart without error.

Deadlock-Free Resource Allocation

Summarize Sign in

Available resources (comma separated):  
3,3

Add Processor Resource Needs:  
e.g. 3,2

Add Processor

Added Processors (Needs):  
• P1: 1,2  
• P2: 2,2

Run Safe Allocation

Reset

Processor 1    Processor 2    Resources

Initial Resources: 3, 3

P1: Accessing resources...  
P1 used resources. Remaining available: 2, 1  
**P1 released resources.** Now available: 3, 3

P2: Accessing resources...  
P2 used resources. Remaining available: 1, 1  
**P2 released resources.** Now available: 3, 3

Summarize Sign in

e.g. 3,2

Add Processor

Added Processors (Needs):  
• P1: 1,2  
• P2: 2,2

Run Safe Allocation

Reset

Processor 1    Processor 2    Resources

Initial Resources: 3, 3

P1: Accessing resources...  
P1 used resources. Remaining available: 2, 1  
**P1 released resources.** Now available: 3, 3

P2: Accessing resources...  
P2 used resources. Remaining available: 1, 1  
**P2 released resources.** Now available: 3, 3

Processor	Res1	Res2	Status
P1	1	2	Executed
P2	2	2	Executed

## Deadlock-Free Resource Allocation

Available resources (comma separated):

3,3

**⚠ Processor needs exceed available resources at Resource 1.**  
Add Processor Resource Needs:

5,5

Add Processor

Added Processors (Needs):

- P1: 1, 2

Run Safe Allocation

**THANK YOU**