# Management of Parts using Data Structures (Stacks, Queues, Linked Lists)

Name: Premkumar S

Roll Number: ME24B1021

Date: 28-04-2025

## Problem Statement

In industries like manufacturing and maintenance, it is crucial to manage various machine parts effectively — including normal, urgent, leaky, fixed, and high-priority parts. This assignment focuses on developing a system that can:  
  
- Add, remove, and manage parts.  
- Handle urgent/emergency parts separately.  
- Track the usage of the system (logging).  
- Manage faulty (leaky) parts and their fixing status.  
- Prioritize important parts efficiently.

## Objectives

- Implement different data structures like Queue, Stack, and Linked Lists to simulate real-world part management.  
- Design modular and efficient code for each operation.  
- Maintain logs and ensure all parts are properly handled according to their priority or status.  
- Test the system using sample parts and display outputs clearly.

## Design Explanation

The solution uses modular functions for each part of the system (queue handling, stack operations, linked list management, logging, etc.) to keep it organized and easy to maintain.

Each category of parts is handled differently:

* Normal requests -> Queue (FIFO)  
  Urgent parts -> Stack (LIFO)  
  Leaky parts -> Singly Linked List  
  Fixed parts -> Doubly Linked List  
  Priority parts -> Circular Linked List  
  Usage Log -> Array

## Why These Data Structures Were Chosen

- Queue ensures fair processing.  
- Stack handles most recent emergencies first.  
- Singly Linked List is efficient for dynamic insertion/deletion.  
- Doubly Linked List helps in forward/backward traversal of fixed parts.  
- Circular Linked List keeps priority parts in continuous rotation.  
- Array for usage log simplifies storage and access.

## Logic of the Code

1. Initialize queue, stack, and linked lists.  
2. Add/remove parts from queue and stack.  
3. Log actions in usage log.  
4. Manage leaky parts using singly linked list.  
5. Manage fixed parts with doubly linked list.  
6. Manage priority parts using circular linked list.  
7. Test the system with sample parts.  
8. Display outputs for each operation.

## Variables and Functions Used

Key Variables:  
- PartQueue pqueue: Normal parts queue  
- UrgentStack estack: Urgent parts stack  
- Usage ulog[LOG\_MAX]: Action history log  
- LeakyNode\* leaky: Head of leaky parts  
- FixedNode\* fixed: Head of fixed parts  
- PrioNode\* prio: Circular linked list head  
  
Key Functions:  
- setupQueue, setupStack: Initialize structures  
- addToQ, removeFromQ: Queue operations  
- pushToS, popFromS: Stack operations  
- logUse, showLog: Log actions  
- addLeaky, removeLeaky, showLeaky: Manage leaks  
- addFixed, showFixedFwd, showFixedBwd: Manage fixed parts  
- addPrio, showPrio: Manage priority parts  
- testQueueStack, testLogging, testParts: Testing modules

## Sample Output Screenshot

