Robotic Assembly Line Simulator

# Using Data Structures

Submitted by:

Name: [Your Name]

Roll Number: [Your Roll Number]

Date: [Submission Date]

# 1. Problem Statement & Objectives

## Problem:

Simulate a futuristic car manufacturing plant where robots manage part assembly, storage, and repairs using optimized data structures.

## Key Objectives:

* ✔ Implement a Queue for part delivery
* ✔ Use a Stack for robot arm assembly (LIFO)
* ✔ Manage garage storage with an Array (fixed capacity)
* ✔ Track defects using Singly Linked Lists
* ✔ Handle VIP upgrades with Circular Linked Lists

# 2. Design Explanation

## Data Structures Used

|  |  |  |
| --- | --- | --- |
| Component | Data Structure | Reason |
| Part Delivery | Queue (FIFO) | Ensures first-in-first-out part processing |
| Robot Arm Assembly | Stack (LIFO) | Matches real-world assembly (last part picked = first installed) |
| Garage Storage | Array | Fixed-size storage with O(1) access; oldest car removed when full |
| Defective Cars | Singly Linked List | Efficient insertions/deletions for dynamic repair tracking |
| Repaired Cars | Doubly Linked List | Bidirectional traversal for quality checks |
| VIP Upgrades | Circular Linked List | Continuous priority processing |

# 3. Code Logic

## Step-by-Step Workflow

Part Delivery (Queue):  
 - Parts (Engine, Chassis, etc.) enqueued  
 - Robot arm dequeues parts → pushes to stack

Assembly (Stack):  
 while stack not empty:  
 part = stack.pop()  
 assemble(part)

Garage Storage (Array):  
 - Fixed 8-slot array  
 - Overflow handling:  
 if garage\_full:  
 ship(garage[0]) # Remove oldest  
 shift\_left() # O(n) operation

Defect Tracking:  
 - Defective cars → Singly Linked List  
 - Repaired cars → Doubly Linked List

VIP Upgrades:  
 - Circular Linked List cycles indefinitely:  
 Car1 → Car5 → Car1 → Car5...

# 4. Key Variables & Functions

## Variables:

|  |  |
| --- | --- |
| Name | Purpose |
| conveyorBelt | Queue for incoming parts |
| assemblyStack | Stack for LIFO assembly |
| garage[8] | Fixed-size garage storage |
| defectiveList | Singly linked list for defective cars |
| vipList | Circular list for VIP cars |

## Functions:

|  |  |
| --- | --- |
| Function | Action |
| enqueue(part) | Adds part to conveyor belt |
| push(part) | Adds part to assembly stack |
| addToGarage(car) | Manages garage overflow |
| moveToRepaired(car) | Transfers car to repaired list |

# 5. Sample Output

(Paste screenshot here showing:)  
- Part assembly sequence  
- Garage overflow handling  
- VIP upgrade cycles

# 6. Viva Preparation

## Expected Questions:

❓ Why use a Stack for assembly?

✅ Answer: Ensures structural integrity - heavy base parts (engine/chassis) are installed first.

❓ Time complexity of garage overflow handling?

✅ Answer: O(n) due to array shifts, but n=8 → practically O(1).

❓ Alternative to Array for garage?

✅ Answer: Circular Queue (but Array simplifies oldest-car removal logic).