# Robotic Assembly Line Simulator

## Data Structures Implementation

Submitted by:

* Name: MOHAMMED MISHAL P
* Roll Number: ME24I1014
* Date: 15/04/2025

## Problem Statement & C Implementation

### Key Objectives:

✔️ Implement part delivery using Queue (FIFO)

✔️ Simulate robot arm with Stack (LIFO)

✔️ Manage garage with Array (fixed 8-slot capacity)

✔️ Track defects using Singly Linked Lists

✔️ Handle VIP cars with Circular Linked Lists

## C Data Structures Design

|  |  |  |
| --- | --- | --- |
| Component | C Implementation | Why Chosen? |
| Part Delivery | Queue struct with front/rear | Natural FIFO processing |
| Robot Arm | Stack struct with top index | LIFO matches assembly order |
| Garage | char\* garage[8] array | Simple fixed-size storage |
| Defective Cars | Singly linked list | Efficient dynamic insertions/deletions |
| VIP Cars | Circular linked list | Continuous priority processing |

Memory Management Note:  
All linked lists use malloc()/free() for dynamic allocation.

## Code Walkthrough

### Key Struct Definitions:  
// Queue for part delivery  
typedef struct {  
 char\* parts[MAX\_PARTS];  
 int front, rear;  
} Queue;  
  
// Stack for robot arm  
typedef struct {  
 char\* parts[MAX\_PARTS];  
 int top;  
} Stack;  
  
// Singly linked list for defects  
typedef struct SingleNode {  
 char\* car;  
 struct SingleNode\* next;  
} SingleNode;  
  
// Circular list for VIP  
typedef struct CircularNode {  
 char\* car;  
 struct CircularNode\* next;  
} CircularNode;

### Critical Functions:

1. Part Delivery (Queue):

void enqueue(Queue\* q, char\* part) {  
 if (q->rear == MAX\_PARTS-1) return;  
 q->parts[++q->rear] = part;  
 }

1. Robot Arm (Stack):

void push(Stack\* s, char\* part) {  
 s->parts[++s->top] = part;  
 }

1. Garage Overflow:

void addToGarage(char\* car) {  
 if (garageCount == 8) {  
 // Shift left to remove oldest  
 for (int i = 0; i < 7; i++)   
 garage[i] = garage[i+1];  
 }  
 garage[garageCount++] = car;  
 }

## 4. Variables & Functions

### Key Global Variables:

char\* garage[8]; // Fixed-size garage  
int garageCount = 0; // Current cars in garage  
SingleNode\* defectiveList; // Head of defect list  
CircularNode\* vipList; // Head of VIP list

### Function Reference Table:

|  |  |  |
| --- | --- | --- |
| Function | Parameters | Action |
| enqueue(Queue\*, char\*) | Queue, part name | Adds part to conveyor belt |
| dequeue(Queue\*) | Queue | Removes part for processing |
| addToDefective(char\*) | Car name | Adds car to defect list (malloc used) |
| moveToRepaired(char\*) | Car name | Transfers car between lists |
| traverseVIP(int) | Number of rounds | Cycles through VIP cars |

## Expected Output

Robot arm picked: Engine  
Robot arm picked: Chassis  
...  
Assembling: Hood  
Assembling: Battery  
...  
Garage full! Shipping out oldest: Car1  
Added to defective list: Car3  
Moved to repaired list: Car3  
VIP Upgrade Cycle: Car1 -> Car5 -> Car1...





