**CARGO DRONE TRAFFIC CONTROLLER**

# R SARAVANAN ME24I1006

15/04/2025

## ASSIGNMENT REPORT :

### Problem statement :

Design a **Cargo Drone Traffic Controller** in C to manage drone deliveries using appropriate data structures. Simulate delivery requests using a queue, handle urgent dispatches via a stack, log completed deliveries in a fixed-size array, and manage drone maintenance using singly and doubly linked lists. Additionally, implement emergency rerouting with a circular linked list. The system should be menu-driven and allow continuous operation until the user chooses to exit.

### Key objectives :

* Use a queue to handle normal drone delivery requests (FIFO).
* Use a stack to manage urgent deliveries in LIFO order.
* Maintain a fixed-size array as a flight log, replacing old entries when full.
* Track overloaded drones using a singly linked list and shift them to a doubly linked list after servicing.
* Handle emergency rerouting with a circular linked list and demonstrate traversal.

### Data Structures Used :

#### Queue ( Array ) :

Storing and managing normal delivery requests in First-In-First-Out (FIFO) order.

#### Stack ( Array ) :

Handling urgent or priority deliveries in Last-In-First-Out (LIFO) order after dequeuing from the request queue.

#### Array :

Maintaining a fixed-size flight log of recently completed deliveries (maximum 6 entries).Oldest entries are removed when new ones exceed the limit.

#### Singly linked list :

Storing overloaded drones that need maintenance or recalibration.

#### Doubly Linked List :

Keeping track of serviced drones, allowing traversal in both forward and backward directions.

#### Circular Linked List :

Managing drones that need emergency rerouting, supporting continuous traversal for circular navigation paths.

### Why has the Data Structure been used and how do they help solve the problem?

1. Queue – For delivery requests

Why: Requests should be handled in the order they come.

Helps by: Making sure no delivery is skipped (first come, first served).

2. Stack – For urgent deliveries

Why: Urgent items need to be sent out quickly.

Helps by: Sending the most recent urgent item first.

3. Array – For completed delivery log

Why: Only the last few deliveries need to be remembered.

Helps by: Keeping a short, simple list of recent deliveries.

4. Singly Linked List – For overloaded drones

Why: To store drones that have issues.

Helps by: Adding and tracking faulty drones easily.

5. Doubly Linked List – For serviced drones

Why: To store drones that got repaired.

Helps by: Allowing you to check in both directions (forward and backward).

6. Circular Linked List – For emergency rerouting

Why: Some drones keep flying in a loop until cleared.

Helps by: Going through the list again and again without stopping.

### Variables used :

Queue Variables

deliveryQueue[6][20] – A 2D array used to store names of regular delivery items.

deliveryFront – Index pointing to the front of the queue (next item to be delivered).

deliveryRear – Index pointing to the rear of the queue (last added delivery).

Stack Variables

urgentStack[6][20] – A 2D array that holds names of urgent deliveries.

urgentTop – Stack pointer that indicates the current top element in the stack (used for LIFO operations).

Flight Log Variables

flightLog[6][20] – A circular-style array that stores the IDs or names of the most recent deliveries.

flightLogCount – Integer that tracks how many deliveries are currently logged.

Singly Linked List (Overloaded Drones)

struct OverloadNode – A node structure with a name field and a next pointer, used to represent overloaded drones.

overloadHead – A pointer to the head node of the singly linked list for overloaded drones.

Doubly Linked List (Serviced Drones)

struct ServicedNode – A node structure that contains name, prev, and next pointers, representing a serviced drone.

servicedHead – Points to the head node of the doubly linked list for drones that have been serviced.

Circular Linked List (Emergency Drones)

struct EmergencyNode – A node with a name and a next pointer, used for drones that are rerouted in emergencies.

emergencyCircular – A pointer to the last node in the circular linked list, forming a loop for continuous traversal.

General Variables in Main

choice – Stores the user's menu selection.

input[20] – A temporary array to hold user input strings.

Temporary pointers like cur, prev, t, d, last – Used in traversing and modifying linked lists.

### FUNCTIONS USED

enqueue(char[]) – Adds a new delivery item to the rear of the queue.

dequeue() – Removes and returns the delivery item from the front of the queue.

push(char[]) – Pushes an urgent delivery onto the top of the stack.

pop() – Pops the top delivery from the stack and returns it.

logDelivery(char[]) – Adds a delivery ID to the flight log, maintaining recent 6 entries using a shifting mechanism.

addOverload(char[]) – Adds a new overloaded drone to the beginning of the singly linked list.

moveToServiced(char[]) – Moves a drone from the overloaded list to the serviced list (doubly linked list).

showServiced() – Displays all serviced drones in both forward and reverse order using doubly linked list traversal.

addEmergency(char[]) – Adds a new emergency rerouted drone into the circular linked list.

showEmergency(int) – Traverses and prints the circular list for the specified number of rounds.

## Output :

