National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures		Course Code: CS 2001
Program:	BS(CS)	Semester:	FALL-2025
Due Date:	12-Sept-2025	Total Marks:	10
Type:	Assignment 1	Page(s):	8
Course Instructor	Rana Waqas Ali	TA.	Syed Aoun Haider Sherazi

Important Instructions:

1. Submit your .cpp files named as your roll number+QuestionNo+FileNo. Dont submit zip files.
2. You are not allowed to copy solutions from other students. We will check your code for plagiarism using plagiarism checkers. If any sort of cheating is found, negative marks will be given to all students involved.
3. Late submission of your solution is not allowed.
4. All questions / parts must be done keeping in mind the time constraint of that Data Structures or marks will be deducted.
5. In case of any confusion, feel free to discuss with Instructor or TA.
6. Do proper edge cases handling for all questions.

Data Structures – Linked List Assignment

Scenario:

You have been hired as a **Software Developer Intern** at **RideNow**, a new ride-hailing startup (similar to Uber/Careem). Your team has been asked to develop the **backend data management system** to store and manage rides using **Linked Lists** (without using STL containers such as list or vector).

The system should manage the following:

- **Ride Requests** made by customers
- **Driver Assignments**
- **Ride History** (Completed and Canceled rides)

Your task is to implement this system using **singly linked lists** (or doubly linked lists where required).

National University of Computer and Emerging Sciences, Lahore Campus

---

Requirements:

1. Ride Request Management

Each ride request has:

- Request ID (unique integer)
- Customer Name (string)
- Pickup Location (string)
- Drop-off Location (string)
- Estimated Fare (float)

You must:

- Add new ride requests at the **end** of the linked list.
- Display all current ride requests.
- Delete a request by ID (if the customer cancels).

---

2. Driver Assignment

When a driver accepts a ride:

- Remove the ride request from the **requests list**.
- Insert it into the **active rides list**.

Each **active ride** must store:

- Ride ID
- Customer Name
- Driver Name
- Pickup Location
- Drop-off Location
- Fare

National University of Computer and Emerging Sciences, Lahore Campus

3. Ride Completion & History

When a ride ends:

- Remove it from the active rides list.
- Insert it into the **ride history list**.
- Store whether it was **Completed** or **Canceled**.

You must also:

- Display **all ride history** in order of completion.
- Search for rides by Customer Name (return all rides of that customer).
- Calculate the **total revenue earned** (sum of fares from completed rides).

---

4. Advanced Functionalities

Implement the following using linked list manipulation:

1. **Sorting Rides by Fare**: Allow the admin to sort completed rides in descending order of fare.
2. **Reversing Ride History**: Allow the admin to view ride history in reverse order (most recent first).
3. **Detect Duplicate Ride Requests**: If the same customer requests multiple identical rides (same pickup/drop), merge them into a single request.
4. **Delete Rides Before a Given Date** (extra challenge: add a timestamp field).
5. **Find the Longest Ride** (based on character length of pickup → drop-off path).

Example Menu (Students must implement a menu-driven program):

1. Add Ride Request

2. Cancel Ride Request

3. Assign Ride to Driver

4. Complete Ride

5. Display Current Ride Requests

6. Display Active Rides
7. Display Ride History
8. Search Rides by Customer
9. Calculate Total Revenue
10. Sort Ride History by Fare
11. Reverse Ride History
12. Detect and Merge Duplicate Ride Requests
13. Delete Old Rides by Date
14. Find Longest Ride
15. Exit

Constraints:

- Must use **Linked Lists** (no arrays, no STL).
- Code must be **modular** (use classes & functions).
- Each operation should be demonstrated clearly.
- Bonus: Implement with **Doubly Linked List** for history to allow forward & backward traversal.

Deliverables:

1. Fully functional C++ program.
2. Test cases with at least **10 ride requests** and **5 drivers**.

Skeleton Code (C++)

Below is the Skeleton Code provided for assignment. You may change it if you require or want to add more stuff in the Menu.

```
#include <iostream>
#include <string>
using namespace std;
```

National University of Computer and Emerging Sciences, Lahore Campus

```
// Node structure for Ride Requests
struct RideRequest {
    int requestID;
    string customerName;
    string pickup;
    string dropoff;
    float fare;
    RideRequest* next;
};

// Node structure for Active Rides
struct ActiveRide {
    int rideID;
    string customerName;
    string driverName;
    string pickup;
    string dropoff;
    float fare;
    ActiveRide* next;
};

// Node structure for Ride History
struct RideHistory {
    int rideID;
    string customerName;
    string driverName;
    string pickup;
    string dropoff;
    float fare;
    string status; // Completed or Canceled
    RideHistory* next;
};

class RideNowSystem {
private:
    RideRequest* requestHead;
    ActiveRide* activeHead;
    RideHistory* historyHead;

public:
    RideNowSystem() {
        requestHead = nullptr;
        activeHead = nullptr;
        historyHead = nullptr;
    }

    // ========== Ride Request Functions ==========
    void addRideRequest(int id, string cname, string pickup, string drop,
float fare);
    void cancelRideRequest(int id);
    void displayRideRequests();

    // ========== Active Ride Functions ==========
```

National University of Computer and Emerging Sciences, Lahore Campus

```
    void assignRideToDriver(int id, string driverName);
    void displayActiveRides();

    // ========== Ride History Functions ==========
    void completeRide(int id, bool completed);
    void displayRideHistory();
    void searchRideByCustomer(string cname);
    void calculateTotalRevenue();

    // ========== Advanced Functionalities ==========
    void sortRideHistoryByFare();
    void reverseRideHistory();
    void detectAndMergeDuplicateRequests();
    void deleteOldRidesByDate(string date); // Extra challenge
    void findLongestRide();
};

int main() {
    RideNowSystem system;
    int choice;

    do {
        cout << "\n====== RideNow System Menu ======\n";
        cout << "1. Add Ride Request\n";
        cout << "2. Cancel Ride Request\n";
        cout << "3. Assign Ride to Driver\n";
        cout << "4. Complete Ride\n";
        cout << "5. Display Current Ride Requests\n";
        cout << "6. Display Active Rides\n";
        cout << "7. Display Ride History\n";
        cout << "8. Search Rides by Customer\n";
        cout << "9. Calculate Total Revenue\n";
        cout << "10. Sort Ride History by Fare\n";
        cout << "11. Reverse Ride History\n";
        cout << "12. Detect & Merge Duplicate Ride Requests\n";
        cout << "13. Delete Old Rides by Date\n";
        cout << "14. Find Longest Ride\n";
        cout << "15. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch(choice) {
            case 1: {
                int id; string cname, pickup, drop; float fare;
                cout << "Enter ID, Customer Name, Pickup, Drop, Fare: ";
                cin >> id >> cname >> pickup >> drop >> fare;
                system.addRideRequest(id, cname, pickup, drop, fare);
                break;
            }
            case 2: {
                int id;
                cout << "Enter Ride ID to cancel: ";
                cin >> id;
```

National University of Computer and Emerging Sciences, Lahore Campus

```
                system.cancelRideRequest(id);
                break;
            }
            case 3: {
                int id; string dname;
                cout << "Enter Ride ID and Driver Name: ";
                cin >> id >> dname;
                system.assignRideToDriver(id, dname);
                break;
            }
            case 4: {
                int id; bool completed;
                cout << "Enter Ride ID and (1 for Completed, 0 for
Canceled): ";
                cin >> id >> completed;
                system.completeRide(id, completed);
                break;
            }
            case 5: system.displayRideRequests(); break;
            case 6: system.displayActiveRides(); break;
            case 7: system.displayRideHistory(); break;
            case 8: {
                string cname;
                cout << "Enter Customer Name: ";
                cin >> cname;
                system.searchRideByCustomer(cname);
                break;
            }
            case 9: system.calculateTotalRevenue(); break;
            case 10: system.sortRideHistoryByFare(); break;
            case 11: system.reverseRideHistory(); break;
            case 12: system.detectAndMergeDuplicateRequests(); break;
            case 13: {
                string date;
                cout << "Enter Date (DD/MM/YYYY): ";
                cin >> date;
                system.deleteOldRidesByDate(date);
                break;
            }
            case 14: system.findLongestRide(); break;
            case 15: cout << "Exiting...\n"; break;
            default: cout << "Invalid choice!\n";
        }
    } while(choice != 15);

    return 0;
}
```

National University of Computer and Emerging Sciences, Lahore Campus