CS/CE 224/272 - Object Oriented Programming and Design Methodologies: Assignment #1

Fall Semester 2025

Due on September 13, 2025, 11:59pm

Student Name, ID, Lecture Section

Instructions

1. This homework consists of one large programming exercise.
2. The objective is to simulate a simple online ride booking system using C++ structs and arrays of structures.
3. You are required to apply structs, menu-driven programming, and basic validation techniques as covered up to Lab 03.
4. Skeleton code is provided in `RideBooking.cpp`. You must complete the given tasks within this file.
5. Submit your solution as a single `.cpp` file, with your name, ID, and section clearly marked in the source code.
6. No extensions will be given. This is an **individual** assignment.
7. Plagiarism or copying will result in a grade of zero, and may be reported to the University Conduct Committee.

Problem 1

(100 points) [**Ride Booking Simulation**] Write a program in C++ that simulates a simple online ride booking system. The system must allow operations for two roles: **Riders** and **Drivers**. All rides must be represented using a `struct` and stored in an array of rides.

Ride Structure

Each ride must include the following information (see Figure **??**).

Field	**Description**
Rider Name	A string between 3 and 30 characters.
Ride ID	A unique 6-digit number (100001–999999).
Driver Name	A string between 3 and 30 characters.
Pickup Location	A string (e.g., “University”, “Home”).
Drop-off Location	A string.
Distance	A `double` representing distance in kilometers.
Fare	A `double` representing fare in PKR.
Ride Status	One of “Ongoing”, “Completed”, or “Cancelled”.

Figure 1: Details of a Ride

At program start, an array of rides and an array of driver names has been declared:

```
Ride rideDetails[100]; string Drivers[50];
```

This limits the system to a maximum of 100 rides and 50 drivers.

System Flow: At launch, the program displays the following menu:

```
Welcome to the Ride Booking Simulation program!  Are you a Rider (1) or a Driver (2)?  Enter
your role:
```

If an invalid role is entered, display: `"Invalid option!  Try again"`.
After selecting a role, the user is asked to enter their name:

```
Please enter your name:
```

Rider Operations: Once logged in as a Rider, the following menu is displayed:

```
Welcome <Rider Name>.  Please Select an Option 1.  Book a Ride 2.  View My Rides 3.  Cancel a
Ride 4.  Return to Main Menu
```

1. **Book a Ride:** Function: `BookRide(name)` Prompts the user for pickup location, drop-off location, and distance. The fare is calculated using `GetFare(distance)` with the scheme shown below:

Problem 1 continued on next page. . .

Distance	Fare Formula
< 2 km	$50 + (50 \times distance)$
2–5 km	$150 + (80 \times (distance - 2))$
> 5 km	$390 + (100 \times (distance - 5))$

The rider selects a driver. A ride struct with status = `Ongoing` is created and added to the array. If no drivers are available, a ride with status `Cancelled` is created.

2. **View My Rides:** Function: `ViewRides(riderName, rideDetails)` Displays all rides associated with the rider.
3. **Cancel a Ride:** Function: `changeStatus(riderName, rideDetails)` Shows all ongoing rides of the rider. The rider enters a Ride ID, which is marked as `Cancelled`.
4. **Return to Main Menu:** Returns the user to the role selection screen.

Driver Operations: If logged in as a Driver, first check if the driver's name exists in the driver list. If not, add it. Then display:

```
Welcome <Driver Name>.  Please Select an Option 1.  View Assigned Rides 2.  Mark Ride as
Completed 3.  View All Rides 4.  Calculate Total Fare 5.  Return to Main Menu
```

1. **View Assigned Rides:** Displays all ongoing rides assigned to this driver.
2. **Mark Ride as Completed:** Function: `changeStatus(driverName, rideDetails)` Shows ongoing rides of the driver and updates chosen ride to `Completed`.
3. **View All Rides:** Displays all rides for this driver, regardless of status.
4. **Calculate Total Fare:** Function: `CalculateTotal(driverName)` Computes total fare earned by the driver across completed rides.
5. **Return to Main Menu:** Returns the user to the role selection screen.

Notes

1. You must use a `struct Ride` to represent rides.
2. Maintain an array of rides to simulate multiple ride records.
3. Validate inputs wherever necessary (e.g., valid menu options, non-negative fare).
4. Ride IDs must be generated incrementally, starting from 100001.
5. Use clear naming conventions (either snake_case or CamelCase).
6. Comment your code clearly and include your name, ID, and section at the top.

Points Distribution

Component	**Points**
IsAvailable Function	10
GetFare Function	10
BookRide Function	20
ViewRides Function	10
ChangeStatus Function	10
CalculateTotal Function	10
Main Menu – Rider	15
Main Menu – Driver	15
TOTAL	**100**