🚌 Bus Reservation & Passenger Management System

(Complete Structure-Based Project)

🎯 Objective

Design and implement a **Bus Reservation & Passenger Management System** using **C++ structures (`struct`) only**.

The system must be **menu-driven**, use **static arrays**, and simulate real-world bus booking operations.

⚠ Constraints

- ☑ Multiple structures\
- ☑ Nested structures\
- ☑ Array of structures\
- ☑ Functions\
- ☑ Searching\
- ☑ Sorting\
- ☑ Deletion using shifting logic\
- ☑ Real-world booking logic

📦 System Requirements

The system should manage:

- Maximum **10 buses**
- Maximum **100 passengers**
- Maximum **200 tickets**

Use static arrays to store data.

🏗 Required Structures

1️⃣ Date Structure

Store travel date.

Members: - day - month - year

2️⃣ Bus Structure

Members: - busNumber - source - destination - totalSeats - availableSeats - farePerSeat

3 Passenger Structure

Members: - passengerID - name - age - gender

4 Ticket Structure (Nested Structure Required)

Members: - ticketID - passenger (full Passenger structure OR passengerID reference) - busNumber - travelDate (Date structure inside) - numberOfSeatsBooked - totalFare

Menu Driven System

Your program must continuously display the following menu until Exit is chosen:

1. Add New Bus\
2. Display All Buses\
3. Register New Passenger\
4. Display All Passengers\
5. Book Ticket\
6. Cancel Ticket\
7. Search Passenger by ID\
8. Search Bus by Bus Number\
9. Sort Passengers by Name\
10. Sort Buses by Available Seats\
11. Display All Tickets\
12. Show Total Revenue\
13. Exit

Functional Requirements

1 Add New Bus

- Store bus details in bus array.
- Initialize `availableSeats = totalSeats`.

2 Register New Passenger

- Add passenger record to passenger array.

3 Book Ticket

- Search bus by busNumber.
- Search passenger by passengerID.
- Ask number of seats.
- Check if enough seats available.

- Reduce availableSeats.
- Generate unique ticketID automatically.
- Calculate totalFare.
- Increase system totalRevenue.

4 Cancel Ticket

- Search ticket by ticketID.
- Increase bus availableSeats.
- Decrease totalRevenue.
- Delete ticket using shifting logic in array.

5 Searching

Implement: - Linear search for passenger by ID. - Linear search for bus by bus number. - Linear search for ticket by ticket ID.

6 Sorting

Implement: - Sort passengers alphabetically by name. - Sort buses by availableSeats (ascending or descending).

Do NOT use built-in sort functions.

7 Display Functions

- Display all buses
- Display all passengers
- Display all tickets
- Display total revenue

Logic Requirements

Your system must:

- Auto-generate ticket IDs
- Maintain global ticket counter
- Maintain total revenue variable
- Properly shift array elements during deletion
- Use modular functions for each operation

Expected Implementation Features

Your solution should include:

- At least 15--20 functions

- 300+ lines of structured code
- Clear separation of logic
- Proper loop usage
- Proper condition usage
- Use of string comparison
- Use of nested struct access

Bonus Challenge (Optional)

- Add seat class (Sleeper / AC / Non-AC)
- Add filtering by source & destination
- Add daily revenue display
- Add bus-wise revenue tracking
