Problem Statements for Aggregation in C++

Problem 1: **Flight Management System**

Problem Statement:

Design a system to manage a fleet of flights in an airline. Create a `Flight` class that stores information about each flight, such as `flightNumber`, `destination`, and `departureTime`. Use a `FlightManager` class to maintain and manage multiple flights, allowing operations such as adding a flight, displaying all flights, updating flight details, and searching for flights based on flight number or destination.

Sample Input:

```
1. Add Flight
2. Display All Flights
3. Update Flight Destination
4. Search Flight by Flight Number
5. Exit
Enter choice: 1
Enter Flight Number: 101
Enter Destination: New York
Enter Departure Time: 14:30
```

Sample Output:

```
Flight added successfully.
```

Problem 2: **Library Book Management**

Problem Statement:

Develop a program to manage books in a library. Create a `Book` class that stores book details like `bookId`, `title`, and `author`. The `Library` class should aggregate multiple `Book` objects. Implement operations to add a book, display all books, update book information, and search for books by title or author.

Sample Input:

```
1. Add Book
2. Display All Books
3. Update Book Title
4. Search Book by Author
5. Exit
Enter choice: 1
Enter Book ID: 202
Enter Title: The Alchemist
Enter Author: Paulo Coelho
```

Sample Output:

```
Book added successfully.
```

Problem 3: **Hospital Patient Management**

Problem Statement:

Create a system to manage patient records in a hospital. Define a `Patient` class with attributes such as `patientId`, `name`, and `disease`. Use a `Hospital` class to aggregate multiple patients and provide functions to admit a patient, display all patient records, update patient details, and search for patients by name or disease.

Sample Input:

```
1. Admit Patient
2. Display All Patients
3. Update Patient Disease
4. Search Patient by Name
5. Exit
Enter choice: 1
Enter Patient ID: 301
Enter Name: John Doe
Enter Disease: Fever
```

Sample Output:

```
Patient admitted successfully.
```

Problem 4: **University Course Enrollment System**

Problem Statement:

Design a system for managing course enrollments at a university. Create a `Course` class with attributes such as `courseId`, `courseName`, and an array of `Student` objects. Implement operations in a `University` class to add students to courses, display enrolled students for a course, update student details, and search for students by ID or name.

Sample Input:

```
1. Add Student to Course
2. Display All Enrolled Students
3. Update Student Name
4. Search Student by ID
5. Exit
Enter choice: 1
Enter Course ID: CS101
Enter Student ID: 401
Enter Student Name: Alice Johnson
```

Sample Output:

```
Student added to course successfully.
```

Problem 5: **Gym Membership Management**

Problem Statement:

Build a system to manage gym memberships. Create a `Member` class with attributes like `memberId`, `name`, and `membershipType`. The `Gym` class should aggregate multiple `Member` objects. Implement operations to add a member, display all members, update membership details, and search for members by name or membership type.

Sample Input:

```
1. Add Member
2. Display All Members
3. Update Membership Type
4. Search Member by Name
5. Exit
Enter choice: 1
Enter Member ID: 501
Enter Name: Sarah Lee
Enter Membership Type: Premium
```

Sample Output:

```
Member added successfully.
```