**C14 - Lab Experience: Aggregation and Composition in a Course Management System**

**Objective**

This lab will reinforce **aggregation** and **composition** in C++ by modeling a **Section (Course Offering)** that contains **Students (Persons) and a Textbook**. You will:

1. Understand **composition** where a Section *owns* a Textbook.
2. Understand **aggregation** where a Section *references* Person objects in a student roster.
3. Explore the impact of modifying shared objects on the overall system.

---

**Concepts Covered**

- **Aggregation**: The Section class contains a list of Person* objects, meaning it does **not** own the students but only references them. Changes to Person objects outside the class affect all references.
- **Composition**: The Section class contains a Textbook object, meaning it owns the Textbook. If the Section is destroyed, its Textbook is also destroyed.

---

**Instructions**

**Step 1: Implement the Classes**

Modify and complete the following classes:

1. Person (stores a name).
2. Textbook (stores a title).
3. Section:
    - Owns a Textbook (composition).
    - Maintains a list of students (Person*), but does **not** own them (aggregation).

---

**Step 2: Implement the Main Program**

Modify C14-Course-Person-Textbook.cpp to:

1. Create a list of students (Person objects).
2. Assign selected students to a Section object.
3. Change a student's name and observe the effects on the Section and the original student list.
4. Print out data before and after modifications to analyze the **effects of aggregation**.

---

**Expected Behavior**

1. Initially, the Section has a subset of students.
2. When a student's name is updated, it reflects in **both** the Section and the main student list, confirming that Section does not own students (aggregation).
3. The Textbook remains unchanged because it is fully owned by the Section (composition).

**Sample main function**

```
int main()
{
    //Create a list of Person objects
    Person* p1 = new Person("Homer");
```

```
    Person* p2 = new Person("Marge");
    Person* p3 = new Person("Bart");
    Person* p4 = new Person("Lisa");

    vector<Person*> vdb{ p1, p2, p3, p4 };

    //Students
    vector<Person*> vp{ p1, p3 };

    Textbook b("OO using C++");

    Section cs2(b, vp);
    cout << "cs2 before changes " << cs2.toString() << endl;

    //Show database
    cout << "\nDatabase\n" << endl;
    for (Person* p : vdb) { cout << p->toString() << endl; }

    p1->setName("Homero");

    cout << "cs2 After changes " << cs2.toString() << endl;

    //Show database
    cout << "\nDatabase\n" << endl;
    for (Person* p : vdb) { cout << p->toString() << endl; }

    cout << "\nAll Done!\n";
}
```

**Example Output**

```
cs2 before changes  Section [ Book:  Book [title: OO using C++]
        Roster:
         Person[ Name:Homer]
         Person[ Name:Bart]

Database

 Person[ Name:Homer]
 Person[ Name:Marge]
 Person[ Name:Bart]
 Person[ Name:Lisa]
cs2 After changes  Section [ Book:  Book [title: OO using C++]
        Roster:
         Person[ Name:Homero]
         Person[ Name:Bart]

Database

 Person[ Name:Homero]
 Person[ Name:Marge]
 Person[ Name:Bart]
 Person[ Name:Lisa]

All Done!
```

## Class Diagram