Programming Question 1: Using operator overloading

Specification:

This question will be based on Assignment 3, Question 2 –Customer Maintenance application to test the operator overloading. Open Customer Maintenance Application,
In your `CustomerList` class, add the following two methods

```
//Add Customer to the Customer List
public static CustomerList operator + (CustomerList c1, Customer c) {…}

//Remove Customer from the Customer List
public static CustomerList operator - (CustomerList c1, Customer c) {…}

Modify the customer maintenance form to use these operators instead of Add() and
Remove() methods. Test the application.
```

Programming Question 2: Using LINQ Queries

Specification:

Create an Inventory class which has the following fields

```
        private string itemNo;
        private string description;
        private int qtyOnHand;
        private decimal unitPrice;
```

Add constructor, property and ToString() method into this class.

Create a console application and test the following LINQ queries on an array of Inventory objects and display the results.

1. To sort Inventory objects by ***description***.
2. To select ***description*** and ***unitPrice*** and sort by ***unitPrice.***
3. To select ***description*** and ***total*** (qtyOnHand * unitPrice), sort by ***total***. (Hint: Use ***let*** to create ***total***.)
4. To display inventory objects that with unitPrice between $30 and $90, order by unitPrice in descending order.
5. To display inventory objects that convert the all ***description*** data to upper case, sort by ***description***.

Use the following test data to add into your collection

itemNumber	***description***	***qtyOnHand***	***unitPrice***
A123	Jig saw	15	12.00
B23	Wrench	20	11.00
C112	Hammer	10	8.00
B135	Power saw	8	79.00
C238	Screwdriver	30	9.50

A890	Lawn mower	6	95.00
C290	Electric sander	12	55.00
C100	Sledge Hammer	9	30.50

Zip both solution folders and submit them in Eagle online for grading.