**Custom List Class Project (out of 150 points)**

**User Stories**

The built-in List<T> class is a generic class that acts as a wrapper over the array class. **You cannot use built-in List or Array methods or properties.** For questions regarding *how* to approach a specific feature, please start by referring to the C# List<T> class documentation to get an idea of how the built-in List<T> class handles various situations and methods.

**(20 points)** As a developer, I want to use Test Driven Development (TDD), so that I can write tests for my methods to pass to ensure proper functionality within my application. There needs to be several tests per method.

**(5 points):** As a developer, I want to make good, consistent commits.

**(10 points): As a developer, I want to use a custom-built list class that stores its values in an array, so that I can store any data type in my collection.**

**(10 points): As a developer, I want a read-only Count property implemented on the custom-built list class, so that I can get a count of the number of elements in my custom list class instance.**

**(10 points): As a developer, I want a Capacity property implemented on the custom-built list class, so that I can publicly see the size of my private array.**

**(10 points): As a developer, I want to create a C# indexer so that I can make the objects in my list accessible via index. I want to properly ensure that a user cannot access an out-of-bounds index.**

**(10 points): As a developer, I want the ability to add an object to an instance of my custom-built list class by imitating the C# Add() method.**

**(10 points): As a developer, I want the ability to remove an object from an instance of my custom-built list class by imitating the C# Remove() method.**

**(10 points): As a developer, I want to be able to override the ToString method that converts the contents of the custom list to a string. HINT: You can use .ToString() inside the method for this**

**(10 points):** As a developer, I want to be able to overload the + operator, so that I can add two instances of the custom list class together.

* List<int> one = new List<int>() {1,3,5}; and List<int> two = new List<int>() {2,4,6};
* List<int> result = one + two;
* result has 1,3,5,2,4,6

**(10 points): As a developer, I want to be able to overload the – operator, so that I can subtract one instance of a custom list class from another instance of a custom list class.**

* **List<int> one = new List<int>() {1,3,5}; and List<int> two = new List<int>() {2,1,6};**
* **List<int> result = one - two;**
* **result has 3,5**

**(5 points): As a developer, I want to write documentation in a .txt file that describes the details and functionality of my – operator overload. I want to include details such as “syntax”, “parameters”, “return type”, and an example of it being used, with the output. I want to use the following piece of documentation as a guideline for my own documentation:**

**https://msdn.microsoft.com/en-us/library/cd666k3e%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396**

**(10 points):** As a developer, I want the ability to zip two custom list class instances together in the form of a zipper. An example:

* I have List<int> odd = new List<int>() {1,3,5}; and List<int> even = new List<int>() {2,4,6};
* odd.Zip(even);
* When lists odd and even are zipped together, your new list will contain values 1,2,3,4,5,6

**(10 points):** As a developer, I want the custom list class to be iterable.

**(10 points):** As a developer, I want to use C# best practices, SOLID design principles, and good naming conventions on the project.

**(Bonus 5 points):** As a developer, I want the ability to sort an instance of my custom-built list class. To be eligible for the bonus points, you may not use Array.Sort() that is already built in and you must tell us what sorting algorithm you used.

**(Bonus 5 points):** As a developer, I want the ability to earn bonus points for an EASTER EGG user story, regarding implementing a specific good practice on one of the methods specified in a user story above.

**NOTICE: get your unit tests (test methods) checked off by an instructor before you begin writing your methods to ensure you are on the correct path.**

**Add Method**

**How does it work?**

**First need to make sure a list is created by declaring the list and instantiating the list.**

**But then we will need to:**

**1) Get list count**

**2) Get maximum values for Array**

**3) If the Count of the Array would make it equal to the Max Capacity of the array then we need to create a new array which will double the array size. It will copy all of the values for the old array, and add them in the appropriate indexes of the new array.**

**(10 points):** As a developer, I want the ability to zip two custom list class instances together in the form of a zipper. An example:

* I have List<int> odd = new List<int>() {1,3,5}; and List<int> even = new List<int>() {2,4,6};
* odd.Zip(even);
* When lists odd and even are zipped together, your new list will contain values 1,2,3,4,5,6

**I need to:**

**Get two custom list objects.**

**Create third custom list object.**

**Create Loop to write customList3. This loop will run (customList1.Count + customList2.Count) times.**

**If i%2**

**List One 3 {1 , 3, 5 } List Two 3 { 2, 4, 6}**

**Loop one**

**i**

When we add a new array.

We will need to loop through old array

How does remove method work

We take an array, It has four numbers: { 1, 2, 3, 4 }

We want to remove 1

We loop through and find 1 in the array.

**(10 points):** As a developer, I want to be able to overload the + operator, so that I can add two instances of the custom list class together.

* List<int> one = new List<int>() {1,3,5}; and List<int> two = new List<int>() {2,4,6};
* List<int> result = one + two;
* result has 1,3,5,2,4,6

Get two CustomList customList objects.

Get Count of array 1 = count1

GetCount of array 2 = count2

Add counts to get size of new array following max capacity definition = newCount

Create new array

Loop through first array count1 times and add 1st array values

Loop through second array count2 times and add 2nd array values.

**(10 points):** As a developer, I want to be able to overload the – operator, so that I can subtract one instance of a custom list class from another instance of a custom list class.

* List<int> one = new List<int>() {1,3,5}; and List<int> two = new List<int>() {2,1,6};
* List<int> result = one - two;
* result has 3,5

Get two lists

Loop through 2nd list for every value and see if it exists in 1st list.

If the value exists, remove it from 1st list

If not, proceed to next item in the list

**(5 points):** As a developer, I want to write documentation in a .txt file that describes the details and functionality of my – operator overload. I want to include details such as “syntax”, “parameters”, “return type”, and an example of it being used, with the output. I want to use the following piece of documentation as a guideline for my own documentation:

https://msdn.microsoft.com/en-us/library/cd666k3e%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396

* **Operator Overload**

Allows the ability to subtract one collection of objects from another collection of objects.

C#

public static className <T> operator - (parameters)

{

//Code  
}

#### Parameters

These parameters are the object lists we are manipulating. The first parameter is the list that you ultimately want. The second parameter is the list of object you will be subtracting from the first list.

Example: (CustomList<T> customList1, CustomList<T> customList2)

#### Returns

This method returns a list of objects.

#### Syntax

CustomList<int> customList3 = customList1 - customList2;

*Example*:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CustomListClass

{

class Program

{

static void Main(string[] args)

{

CustomList<int> customList1 = new CustomList<int>();

CustomList<int> customList2 = new CustomList<int>();

customList1.Add(1);

customList1.Add(2);

customList1.Add(3);

customList2.Add(2);

customList2.Add(1);

customList2.Add(6);

CustomList<int> customList3 = new CustomList<int>();

customList3 = (customList1 - customList2);

Console.ReadLine();

}

}

}

This example returns a new customList3 value of { 3, 5 }.

**(Bonus 5 points):** As a developer, I want the ability to sort an instance of my custom-built list class. To be eligible for the bonus points, you may not use Array.Sort() that is already built in and you must tell us what sorting algorithm you used.

Need to look up C# documention on how sorting works. Recommends using Bubble Sort.

**(Bonus 5 points):** As a developer, I want the ability to earn bonus points for an EASTER EGG user story, regarding implementing a specific good practice on one of the methods specified in a user story above.

Need to find 2 examples of static. Need to find a third place that this should be implanted.