**Custom List Class Project (out of 120 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.**

**(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.

**(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 the ability to add an object to an instance of my custom-built list class.

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

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

**(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.

**(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.

**(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.

**(10 points):** As a developer, I want a 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 the ability to Enumerable.Zip() two custom list class instances together.

**(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.

**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.**

**michaelterrill [1:49 PM]**

**- Grade: 56/120 = 47%**

**o TDD: -13 points for only have four tests. Yes, you did one test for each of the features you completed. However, we said many times we expect multiple tests for each method.**

**o -8 points for Remove not working properly. See below.**

**o -5 points for Count being a field instead of a property. Also, just name it count instead of arrayCount.**

**o -10 points for not overloading the + operator**

**o -10 points for not overloading the – operator**

**o -10 points for not having a Zip method to zipper two lists together**

**o -8 points for not overriding the ToString() properly. See below.**

**o**

**- UnitTest1.cs**

**o Make sure to change the name from UnitTest1 to a better, more descriptive class name.**

**o Good job changing the Unit Test project name**

**- The method to override the ToString() should be located underneath the constructor, not above it.**

**- Good job using C# best practices to assign values your member variables in the constructor.**

**- ToString()**

**o You still have code in your method that says “Testing One Tow Three”. This code should have been removed prior to turning in your project.**

**o As it stands, your ToString() doesn’t stringify the contents of your list.**

**o For the ToString(), you can loop through each index of your list (or array) and call .ToString() on the index. That way you are not calling the built-in .ToString() on the object itself, but rather the built-in .ToString() objects located inside the list.**

**- Add()**

**o You did a fantastic job with your logic for the Add() method**

**- Remove()**

**o Your Remove method doesn’t work because the logic inside of if(myItems.Length < arrayCount) doesn’t execute. This is because inside your Add method you increase the size of your increasedTempArray by 4, which then gets set to your myItems array. That means if I have a count of 3, your myItems.Length (which is 4) will not be less than 3. Therefore, the logic will never execute.**

**o After changing your code to get past that point another issue showed up. Since you set your removeTempArray size to 0, it throws an IndexOutOfRange exception because it is checking an array that has a size of 0.**