

Minus Operator Overloading for CustomList class:

Syntax:

- Use the “—” symbol to subtract two instances of a CustomList from each other.
- It doesn't matter what data type the CustomList is.

Parameters:

- You need two different lists to pass in. For example:

-(CustomList<T> name1, CustomList<T> name2)

Return type:

- Returns a new CustomList<T> after subtracting one list from the other

Example:

```
public static CustomList<T> operator - (CustomList<T> list1, CustomList<T> list2)
{
    CustomList<T> newList = new CustomList<T>();
    for(int i = 0; i < list1.Count; i++)
    {
        bool isEqual = false;
        for(int j = 0; j < list2.Count; j++)
        {
            if (list1[i].Equals(list2[j]))
            {
                isEqual = true;
            }
            else if(j == (list2.Count - 1) && isEqual == false)
            {
                newList.Add(list1[i]);
            }
        }
    }
    return newList;
}
```

Example Unit Test:

```
[TestMethod]
public void CustomList_Subtract_SubtractTwoListsIntsTogether()
{
    //arrange
    CustomList<int> list1 = new CustomList<int>();
    CustomList<int> list2 = new CustomList<int>();
    string expected = "1";
}
```

```
int int1 = 1;
int int2 = 2;
int int3 = 3;
int int4 = 2;
int int5 = 2;
int int6 = 3;

//act
list1.Add(int1);
list1.Add(int2);
list1.Add(int3);
list2.Add(int4);
list2.Add(int5);
list2.Add(int6);

CustomList<int> actual = (list1 - list2);

//assert
Assert.AreEqual(expected, actual.ToString());
}
```