Pass task 10.1: Unit Testing

Part 1

```
public void combineArrayStringWithDelimiterTest()
{
    StringUtilities target = new StringUtilities(); // TODO: Initialize to an appropriate value
    string[] stringArray = {"DP1","is","cool"}; // TODO: Initialize to an appropriate value
    string delimiter = " "; // TODO: Initialize to an appropriate value
    string expected = "DP1 is cool"; // TODO: Initialize to an appropriate value
    string actual;
    actual = target.combineArrayStringWithDelimiter(stringArray, delimiter);
    Assert.AreEqual(expected, actual);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}

/// <summary>
///A test for replaceSpacesInStringByHyphens
///</summary>
[TestMethod()]
public void replaceSpacesInStringByHyphensTest()
{
    StringUtilities target = new StringUtilities(); // TODO: Initialize to an appropriate value
    string aString = "DP1 is cool"; // TODO: Initialize to an appropriate value
    string expected = "DP1-is-cool"; // TODO: Initialize to an appropriate value
    string actual;
    actual = target.replaceSpacesInStringByHyphens(aString);
    Assert.AreEqual(expected, actual);
    //Assert.Inconclusive("Verify the correctness of this test method.");
}
```

Figure (1) Both test method for combineArrayStringWithDelimiter function and replaceSpacesInStringByHyphens function

```
public string combineArrayStringWithDelimiter(string[] stringArray, string delimiter)
{
    string str = default(string);
    foreach (string item in stringArray)
    {
        str += item + delimiter;


    }
    return str;
}
```

Figure (2) combineArrayStringWithDelimiter function before fixing it to meet the test criteria.

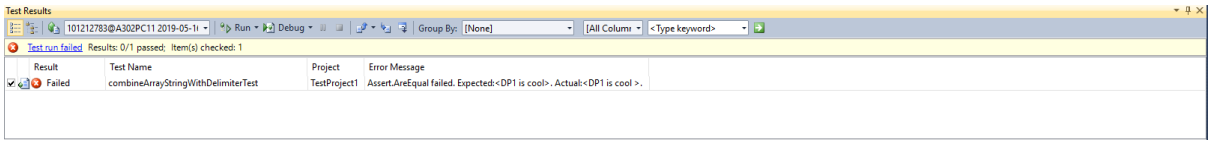

Figure (3) Test error for combineArrayStringWithDelimiter function

```
public string combineArrayStringWithDelimiter(string[] stringArray, string delimiter)
{
    string str = default(string);
    foreach (string item in stringArray)
    {
        str += item + delimiter;


    }
    return str.TrimEnd();
}
```

Figure (4) combineArrayStringWithDelimiter function after fixing it to meet the test criteria.

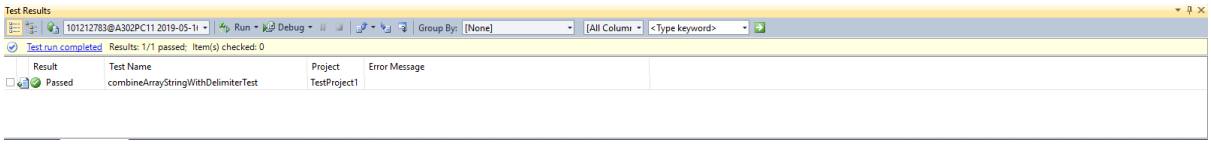

Figure (5) Test passed for combineArrayStringWithDelimiter function

```
public string replaceSpacesInStringByHyphens(string aString)
{
    string[] stringArray = aString.Split(null);
    return combineArrayStringWithDelimiter(stringArray, "-");
}
```

Figure (6) replaceSpacesInStringByHyphens function before fixing it to meet the test criteria.

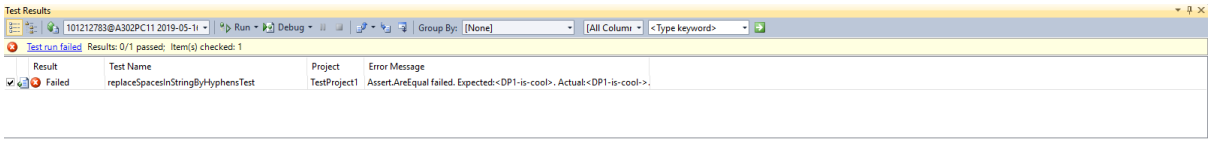

Figure (7) Test error for replaceSpacesInStringByHyphens function

```
public string replaceSpacesInStringByHyphens(string aString)
{
    string[] stringArray = aString.Split(null);
    return combineArrayStringWithDelimiter(stringArray, "-").TrimEnd('-');
}
```

Figure (8) replaceSpacesInStringByHyphens function after fixing it to meet the test criteria.

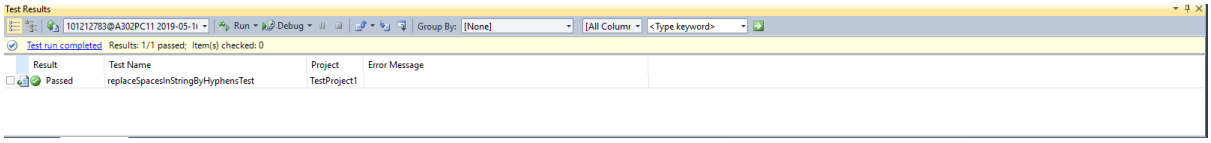

Figure (9) Test passed for replaceSpacesInStringByHyphens function

Part 2

```
namespace Tests
{
    [TestClass()]
    0 references
    public class AttackResultTests
    {
        [TestMethod()]
        | 0 references
        public void AttackResultTest()
        {
            AttackResult attackResult = new AttackResult(ResultOfAttack.GameOver, "Game over", 0, 0);

            string result = attackResult.ToString();
            string outcome = "Game over";
            Assert.AreEqual(result, outcome);
        }
    }
}
```

Figure (10) Unit testing for AttackResult function

For the battleship testing I decided to test whether the AttackResult function is working as intended. In the unit testing class, I called the AttackResult function and assigned the values into a new variable called attackResult where the game is Over so ResultOfAttack.GameOver is passed into the function value and the string “Game over” will appear. I then passed on the value of attackResult into a new string variable called result and added the .ToString() function so that it will convert the values into a string. I then created another string variable called outcome and assigned the string value with “Game over” this is so that I can compare the result of the function and the expected outcome that the function should produce. I then use Assert.AreEqual function to compare both result and outcome. If both variables are equal, then the test is a success otherwise the test is a failure. The result of the test is a success as it is shown the figure (11).

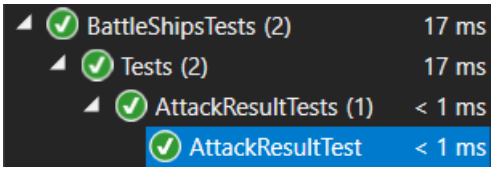

Figure (11) Test success for AttackResultTest Unit testing.