1…**Description**

**Note :**

**Please use only the UnitTest.cs file and do not make any changes to any other files.**

**Objective:** This application aims to educate learners about developing a NUnit test case to verify that the method returns the expected concatenated string based on two input parameters.

Write the testcase using NUnit for the given scenario.

Class "Program" has a method **'Message'**

This method will return the result as string based on the 2 inputs **input1** and **input2.**

In the **UnitTest**class, you are required to create 1 test method for the **Message**method.

|  |  |
| --- | --- |
| **Method** | **Rules** |
| public void Test\_Message\_String() | This test method checks if the '**Message**' method in the **Program** class returns the correct data. |

**Note:**

1) Add the required test attribute for the class UnitTest

2) Add the required test attribute for each test methods

3) In every test methods, you should need to use only one **Assert**

4) Assert whether the actual is **equal**to the expected

Answer:

using NUnit.Framework;

namespace StringConcatenation

{

    [TestFixture] // Add the required test attribute for the class UnitTest

    class UnitTest

    {

        [Test] // Add the required test attribute for the test method

        public void Test\_Message\_String()

        {

            // Arrange

            string input1 = "Hello";

            string input2 = "World";

            string expected = "Hello World";

            // Act

            string actual = Program.Message(input1, input2);

            // Assert

            Assert.AreEqual(expected, actual); // Use only one Assert to check if the actual is equal to the expected

        }

    }

}

2..

**Description**

**Note :**

**Please use only the UnitTest.cs file and do not make any changes to any other files.**

**Objective:** This application aims to educate learners about developing NUnit test cases to verify the behavior of the method. The test cases ensure that the method correctly returns true for valid vowel characters and false for invalid non-vowel characters.

Write the testcase using NUnit for the given scenario.

Class "Program" has a method '**CheckVowels'**

This method will decide the result (true or false) based on the input passed.

In the **UnitTest**class, you are required to create test methods for the **CheckVowels**method.

|  |  |
| --- | --- |
| **Method** | **Rules** |
| public void Test\_CheckVowels\_Valid() | This test verifies the correctness of the **CheckVowels**method when provided with a valid vowel and confirms that it returns **true**. |
| public void Test\_CheckVowels\_Invalid() | This test verifies the correctness of the **CheckVowels**method when provided with an invalid non-vowel character and confirms that it returns **false**. |

**Note:**

1) Add the required test attribute for the class UnitTest

2) Add the required test attribute for each test methods

3) In every test methods, you should need to use only one **Assert**

4) Assert whether the actual is **equal**to the expected

Answer:

using System.Text;

using NUnit.Framework;

namespace VowelChecker

{

    // Add required NUnit test attribute

    [TestFixture]

    class UnitTest

    {

        [Test]

        public void Test\_CheckVowels\_Valid()

        {

            // Arrange

            char input = 'a';

            bool? expected = true;

            // Act

            bool? actual = Program.CheckVowels(input);

            // Assert

            Assert.AreEqual(expected, actual); // Use only one Assert to check if the actual is equal to the expected

        }

        [Test]

        public void Test\_CheckVowels\_Invalid()

        {

            // Arrange

            char input = 'b';

            bool? expected = false;

            // Act

            bool? actual = Program.CheckVowels(input);

            // Assert

            Assert.AreEqual(expected, actual); // Use only one Assert to check if the actual is equal to the expected

        }

    }

}

3…**Description**

**Note :**

**Please use only the UnitTest.cs file and do not make any changes to any other files.**

**Objective:**This application aims to educate learners about NUnit test cases to verify method functionality, ensure correct array addition and expected value return, and validate behavior when a specified value is not found.

Write the testcase using NUnit for the given scenario.

The **Program** class contains a method named '**AddValues**,' which aims to add values to a static integer array (**a1**) and then search for a specified value within that array. The method returns an integer result.

In the **UnitTest**class, you are required to create test methods for the **AddValues**method.

|  |  |
| --- | --- |
| **Method** | **Rules** |
| public void Test\_AddValues\_RightValue() | Tests if the '**AddValues**' method returns the correct value for a specified input.  Validates that the method accurately adds values to the array and returns the expected result. |
| public void Test\_AddValues\_ValueNotFound() | Tests the '**AddValues**' method when the specified value is not found in the array.  Validates the method's behavior in scenarios where the target value is absent, expecting a return value of 0. |

**Note:**

1) Add the required test attribute for the class UnitTest

2) Add the required test attribute for each test methods

3) In every test methods, you should need to use only one **Assert**

4) Assert whether the actual is **equal**to the expected

Answer: using System.Text;

using NUnit.Framework;

namespace ArraySearch

{

    // Add required NUnit test attribute

    [TestFixture]

    class UnitTest

    {

        [Test]

        public void Test\_AddValues\_RightValue()

        {

            // Arrange

            int expected = 5;

            // Act

            int actual = Program.AddValues(5).Value;

            // Assert

            Assert.AreEqual(expected, actual);

        }

        [Test]

        public void Test\_AddValues\_ValueNotFound()

        {

            // Arrange

            int expected = 0;

            // Act

            int actual = Program.AddValues(15).Value;

            // Assert

            Assert.AreEqual(expected, actual);

        }

    }

}

4…**Description**

**Note :**

**Please use only the UnitTest.cs file and do not make any changes to any other files.**

**Objective:** This application aims to educate learners about NUnit test cases designed to verify the functionality of the method. The tests ensure that the method consistently produces a non-null integer array output and accurately multiplies values, returning the expected result.

Write the testcase using NUnit for the given scenario.

The **Program**contains a method named '**Multiply**,' which is designed to return an integer array as a result.

In the **UnitTest**class, you are required to create test methods for the **Multiply**method.

|  |  |
| --- | --- |
| **Method** | **Rules** |
| public void Test\_Multiply\_NotReturningNull() | This test method checks if the '**Multiply**' method does not return a null value.   It ensures that the method consistently generates a valid non-null integer array output. |
| public void Test\_Multiply\_RightValue() | This test method focuses on validating the accuracy of the '**Multiply**' method.  It ensures that the method correctly multiplies values and returns the expected integer array result. |

**Note:**

1) Add the required test attribute for the class UnitTest

2) Add the required test attribute for each test methods

3) In every test methods, you should need to use only one **Assert**

4) Assert whether the actual is **equal**to the expected

 using NUnit.Framework;

using ArrayMultiplication;

namespace ArrayMultiplicationTests

{

    [TestFixture]

    public class UnitTest

    {

        [Test]

        public void Test\_Multiply\_NotReturningNull()

        {

            // Arrange

            int input = 5;

            // Act

            int[] result = Program.Multiply(input);

            // Assert

            Assert.IsNotNull(result);

        }

        [Test]

        public void Test\_Multiply\_RightValue()

        {

            // Arrange

            int input = 5;

            int[] expected = new int[10];

            for (int i = 1; i < 10; i++)

            {

                expected[i] = input \* i;

            }

            // Act

            int[] result = Program.Multiply(input);

            // Assert

            Assert.AreEqual(expected, result);

        }

    }

}

5….**Description**

**Note :**

**Please use only the UnitTest.cs file and do not make any changes to any other files.**

**Objective:** This application aims to educate learners about NUnit test cases designed to verify the functionality of the method. The tests ensure that the method generates a list of student objects with unique values and successfully adds predefined student objects to the list, returning the updated list as expected.

Write the testcase using NUnit for the given scenario.

The 'Program' class contains a method named '**FinalList**'. This method will return the result as List<Student>

In the **UnitTest**class, you are required to create test methods for the **FinalList**method.

|  |  |
| --- | --- |
| **Method** | **Rules** |
| public void Test\_FinalList\_UniqueValues() | Verify that the '**FinalList**' method generates a List of Student objects with only unique values.  This test ensures that each Student in the returned list is distinct, avoiding any duplication. |
| public void Test\_FinalList\_AddingValuesToList() | Validate the functionality of the '**FinalList**' method in adding Student objects to the list.  This test focuses on confirming that the '**FinalList**' method successfully adds predefined Student objects to the list and returns the updated list. |

**Note:**

1) Add the required test attribute for the class UnitTest

2) Add the required test attribute for each test methods

3) In every test methods, you should need to use only one **Assert**

4) Assert whether the actual is **equal**to the expected

Answer:

using System.Collections.Generic;

using NUnit.Framework;

namespace DemoAppCore

{

    // Add required NUnit test attribute

    [TestFixture]

    class UnitTest

    {

        [Test]

        public void Test\_FinalList\_UniqueValues()

        {

            // Arrange

            // Act

            List result = Program.FinalList();

            // Assert

            Assert.AreEqual(result.Count, new HashSet(result).Count);

        }

        [Test]

        public void Test\_FinalList\_AddingValuesToList()

        {

            // Arrange

            List expected = new List

            {

                new Student { Id = 150, Name = "Suresh" },

                new Student { Id = 151, Name = "Karthick" },

                new Student { Id = 152, Name = "Prem" }

            };

            // Act

            List result = Program.FinalList();

            // Assert

            Assert.AreEqual(expected.Count, result.Count);

            for (int i = 0; i < expected.Count; i++)

            {

                Assert.AreEqual(expected[i].Id, result[i].Id);

                Assert.AreEqual(expected[i].Name, result[i].Name);

            }

        }

    }

}