**Task:**

Follow the steps listed below to write the NUnit test cases for the application.

* Create a Unit test project(.Net Framework) in the solution provided.
* Add the CalcLibrary project as reference
* Create a class “CalculatorTests” to write all the test cases for the methods in the solution
* Use the ‘TestFixture’, ‘SetUp’ and ‘TearDown’ attributes, to declare, initialize and cleanup activities respectively
* Create a Test method to check the addition functionality
* Use the ‘TestCase’ attribute to send the inputs and the expected result
* Use Assert.That to check the actual and expected result match

**Code:**

using NUnit.Framework;

using CalcLibrary;

using System;

**namespace** CalcLibraryTests

{

[TestFixture]

**public** **class** CalculatorTests

{

**private** SimpleCalculator \_calculator;

[SetUp]

**public** void SetUp()

{

\_calculator = new SimpleCalculator();

}

[TearDown]

**public** void TearDown()

{

\_calculator = null;

}

[Test]

[TestCase(10, 5, 15)]

[TestCase(-3, -7, -10)]

[TestCase(0, 0, 0)]

**public** void Addition\_WhenCalled\_ReturnsCorrectResult(double a, double b, double expected)

{

**var** result = \_calculator.Addition(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[TestCase(10, 5, 5)]

[TestCase(-3, -7, 4)]

[TestCase(0, 5, -5)]

**public** void Subtraction\_WhenCalled\_ReturnsCorrectResult(double a, double b, double expected)

{

**var** result = \_calculator.Subtraction(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[TestCase(3, 5, 15)]

[TestCase(-3, 2, -6)]

[TestCase(0, 10, 0)]

**public** void Multiplication\_WhenCalled\_ReturnsCorrectResult(double a, double b, double expected)

{

**var** result = \_calculator.Multiplication(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[TestCase(10, 2, 5)]

[TestCase(-8, 2, -4)]

[TestCase(0, 1, 0)]

**public** void Division\_WhenCalled\_ReturnsCorrectResult(double a, double b, double expected)

{

**var** result = \_calculator.Division(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

**public** void Division\_ByZero\_ThrowsArgumentException()

{

**var** ex = Assert.Throws<ArgumentException>(() => \_calculator.Division(5, 0));

Assert.That(ex.Message, Is.EqualTo("Second Parameter Can't be Zero"));

}

[Test]

**public** void AllClear\_WhenCalled\_ResetsResultToZero()

{

\_calculator.Addition(10, 5);

\_calculator.AllClear();

Assert.That(\_calculator.GetResult, Is.EqualTo(0));

}

}

}

**Output:**

