DAY18 ASSIGNMENT
BY
PAVAN KUMAR (16-02-2022)

Q1). What is the use of XML and points discussed in Class.

XML: EXRENSIBLE MARKUP LANGUAGE

- XML is case sensitive.
- XML is most widely-used formats for sharing structured information between programs, between computers and people, both locally and across networks.
- Two kinds of XML are

1) TAG BASED XML

2) ATTRIBUTE BASED XML

- XML used for universal data transfer mechanism to send data across different platforms.
- XML has only one root tag, multiple tags won't work in XML.
- Compared to Tag Based XML Attribute Based XML is easier to store the data because Attribute Based XML takes less memory to store the data.

Q2). Create a simple xml to illustrate: a. Tag based xml with 10 products b. Attribute based xml

TAG BASED XML:

```
<Products>
  <Product1>
    <ID>1</ID>
    <Name>Samsung</Name>
    <Price>25000</Price>
    <Size>30Inch</Size>
   </Product1>

 <Product2>
    <ID>2</ID>
    <Name>Sony</Name>
    <Price>35000</Price>
    <Size>32Inch</Size>
   </Product2>

<Product3>
    <ID>3</ID>
    <Name>LG</Name>
    <Price>15000</Price>
    <Size>24Inch</Size>
   </Product3>

<Product4>
    <ID>4</ID>
```

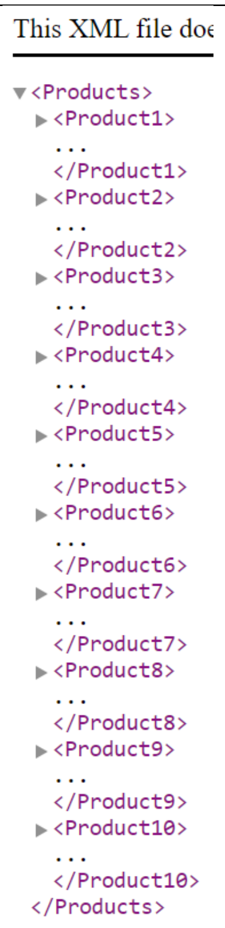
```
     <Name>RealMe</Name>
     <Price>20000</Price>
     <Size>40Inch</Size>
   </Product4>
<Product5>
     <ID>5</ID>
     <Name>MI</Name>
     <Price>30000</Price>
     <Size>43Inch</Size>
   </Product5>

<Product6>
     <ID>6</ID>
     <Name>Medicine</Name>
     <Price>300</Price>
     <Quantity>4Strips</Quantity>
   </Product6>

<Product7>
    <ID>7</ID>
     <Name>Medicine</Name>
     <Price>3000</Price>
     <Quantity>14Strips</Quantity>
   </Product7>

  <Product8>
      <ID>8</ID>
     <Name>Medicine</Name>
     <Price>1300</Price>
     <Quantity>10Strips</Quantity>
   </Product8>

<Product9>
      <ID>9</ID>
     <Name>Medicine</Name>
     <Price>5000</Price>
     <Quantity>20Strips</Quantity>
   </Product9>

<Product10>
      <ID>10</ID>
     <Name>Laptop</Name>
     <Price>13000</Price>
     <Quantity>1</Quantity>
   </Product10>
</Products>
```

ATTRIBUTE BASED XML:

```
<Products>
<Product1>ID="1" Name="Sony" Price="40000" Size="60"></Product1>
<Product2>ID="2" Name="Samsung" Price="41000" Size="55"></Product2>
<Product3>ID="3" Name="MI" Price="42000" Size="50"></Product3>
<Product4>ID="4" Name="RealMe" Price="43000" Size="45"></Product4>
<Product5>ID="5" Name="LG" Price="44000" Size="40"></Product5>
```

```
<Product6>ID="6" Name="Sony" Price="40000" Size="35"></Product6>
<Product7>ID="7" Name="UV" Price="40000" Size="30"></Product7>
<Product8>ID="8" Name="Toshiba" Price="40000" Size="25"></Product8>
<Product9>ID="9" Name="OnePlus" Price="40000" Size="20"></Product9>
<Product10>ID="10" Name="Apple" Price="400000" Size="72"></Product10>
</Products>
```

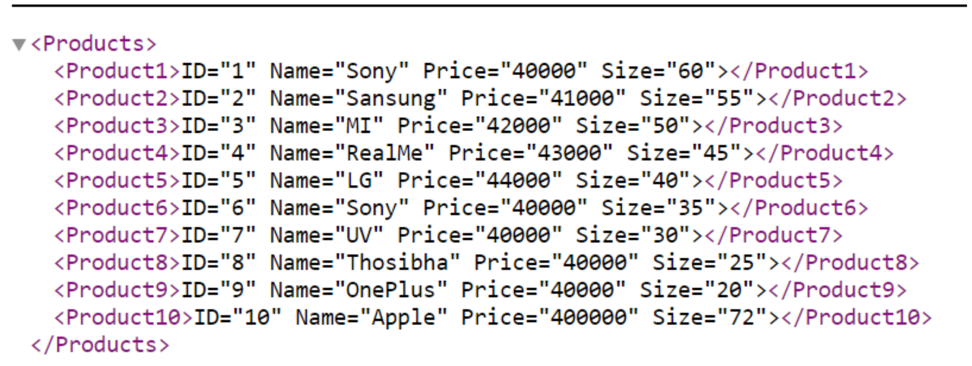
OUTPUT:



This XML file doe

```
▼<Products>
  ▶<Product1>
    ...
    </Product1>
  ▶<Product2>
    ...
    </Product2>
  ▶<Product3>
    ...
    </Product3>
  ▶<Product4>
    ...
    </Product4>
  ▶<Product5>
    ...
    </Product5>
  ▶<Product6>
    ...
    </Product6>
  ▶<Product7>
    ...
    </Product7>
  ▶<Product8>
    ...
    </Product8>
  ▶<Product9>
    ...
    </Product9>
  ▶<Product10>
    ...
    </Product10>
  </Products>
```

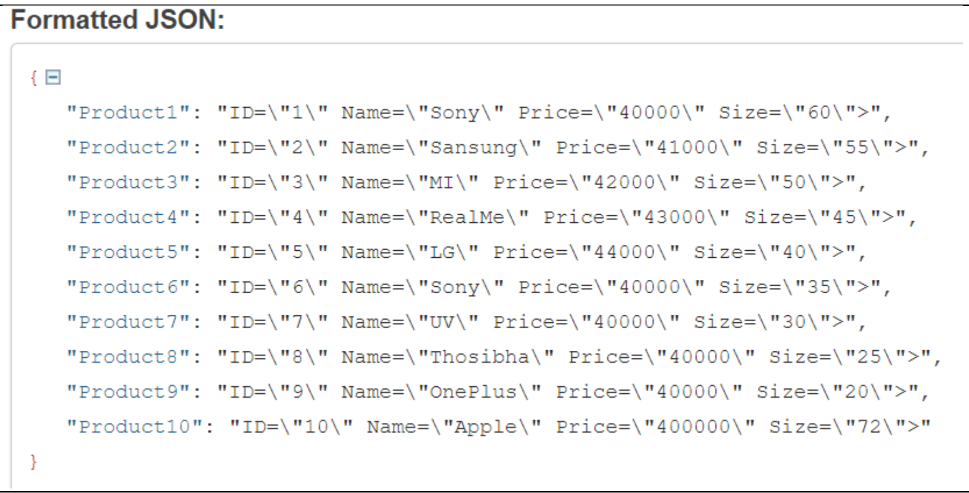
ATTRIBUTE OUTPUT:



```
▼<Products>
   <Product1>ID="1" Name="Sony" Price="40000" Size="60"></Product1>
   <Product2>ID="2" Name="Sansung" Price="41000" Size="55"></Product2>
   <Product3>ID="3" Name="MI" Price="42000" Size="50"></Product3>
   <Product4>ID="4" Name="RealMe" Price="43000" Size="45"></Product4>
   <Product5>ID="5" Name="LG" Price="44000" Size="40"></Product5>
   <Product6>ID="6" Name="Sony" Price="40000" Size="35"></Product6>
   <Product7>ID="7" Name="UV" Price="40000" Size="30"></Product7>
   <Product8>ID="8" Name="Thosibha" Price="40000" Size="25"></Product8>
   <Product9>ID="9" Name="OnePlus" Price="40000" Size="20"></Product9>
   <Product10>ID="10" Name="Apple" Price="400000" Size="72"></Product10>
 </Products>
```

Q3). Convert the above xml to JSON and display the JSON data

OUTPUT FOR THE FORMATTED XML TO JSON DATA:



Formatted JSON:

```
{
   "Product1": "ID=\"1\" Name=\"Sony\" Price=\"40000\" Size=\"60\">",
   "Product2": "ID=\"2\" Name=\"Sansung\" Price=\"41000\" Size=\"55\">",
   "Product3": "ID=\"3\" Name=\"MI\" Price=\"42000\" Size=\"50\">",
   "Product4": "ID=\"4\" Name=\"RealMe\" Price=\"43000\" Size=\"45\">",
   "Product5": "ID=\"5\" Name=\"LG\" Price=\"44000\" Size=\"40\">",
   "Product6": "ID=\"6\" Name=\"Sony\" Price=\"40000\" Size=\"35\">",
   "Product7": "ID=\"7\" Name=\"UV\" Price=\"40000\" Size=\"30\">",
   "Product8": "ID=\"8\" Name=\"Thosibha\" Price=\"40000\" Size=\"25\">",
   "Product9": "ID=\"9\" Name=\"OnePlus\" Price=\"40000\" Size=\"20\">",
   "Product10": "ID=\"10\" Name=\"Apple\" Price=\"400000\" Size=\"72\">"
}
```

Q4). Research and write the benefits of JSON over XML

- JSON: JAVASCRIPT OBJECT NOTATION
- JSON doesn't use **end** tag and it's shorter.
- JSON can use arrays.
- Parsers are less complex, which requires less processing time and memory overhead.

Q6). Create a layered architecture project with separate class library for Business logic.

CODE:

BY CONSOLE APPLICATION:

ALGEBRA CLASS:

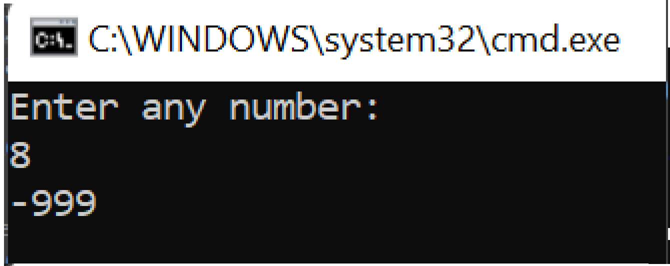
```
using System;
using System.Collections.Generic;

namespace MathematicsLibrary
{
  /// <summary>
  /// DONE BY: PAVAN
  /// PURPOSE: CREATING AN ALGEBRA CLASS;
  /// </summary>
  public class Algebra
   {
    public static int Factorial(int n)
    {
      int fact = 1;
      if (n == 0)
        return 1;
      else if (n > 7)
        return -999;
      else if (n < 0)
        return -9999;
      else
      {
        for (int i = 1; i <= n; i++)
          fact = fact * i;
        return fact;
```

```
            }
        }
    }
}
```

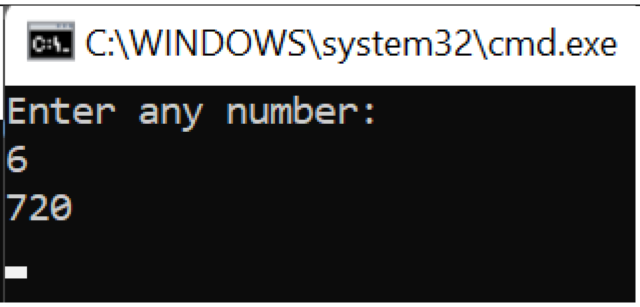
PROJECT CLASS:

```
using System;
using System.Collections.Generic;
using MathematicsLibrary;

namespace Day18Project1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("Enter any number:");
            n= Convert.ToInt32(Console.ReadLine());
            Console.WriteLine(Algebra.Factorial(n));
            Console.ReadLine();
        }
    }
}
```

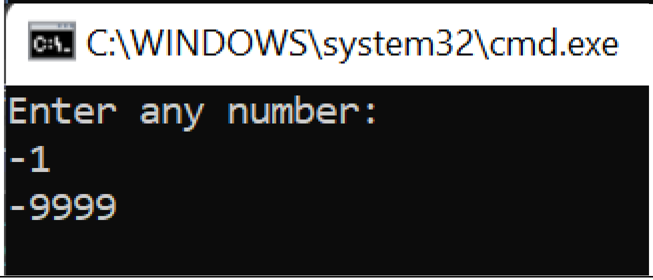
OUTPUT:



```
C:\WINDOWS\system32\cmd.exe
Enter any number:
8
-999
```



```
C:\WINDOWS\system32\cmd.exe
Enter any number:
6
720
```



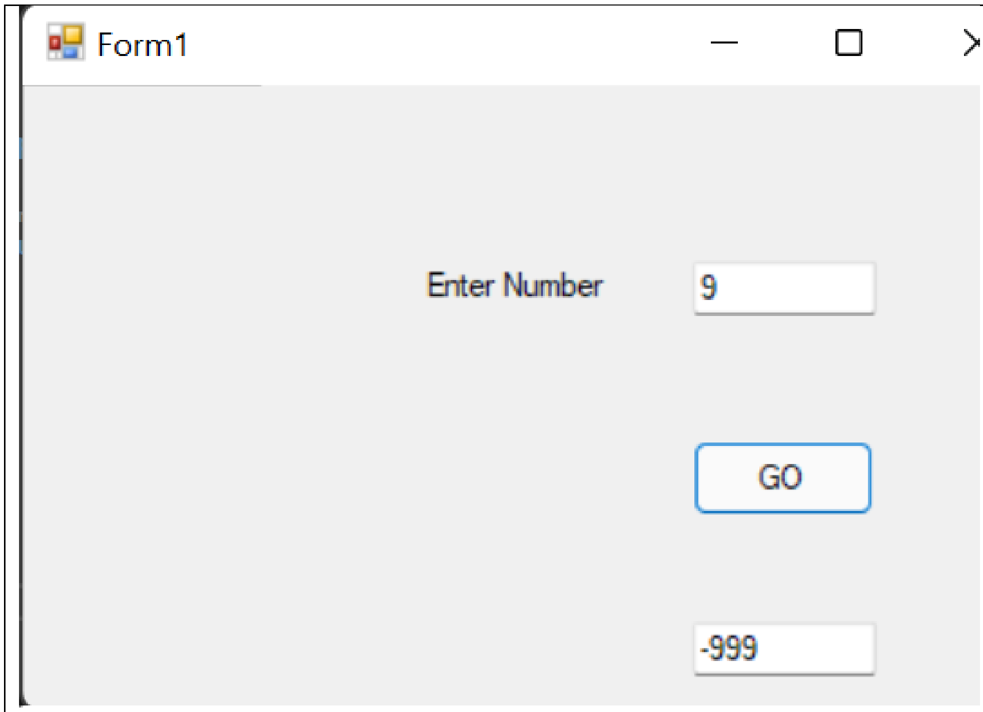
```
C:\WINDOWS\system32\cmd.exe
Enter any number:
-1
-9999
```

BY WINDOWS APPLICATION:

CODE:

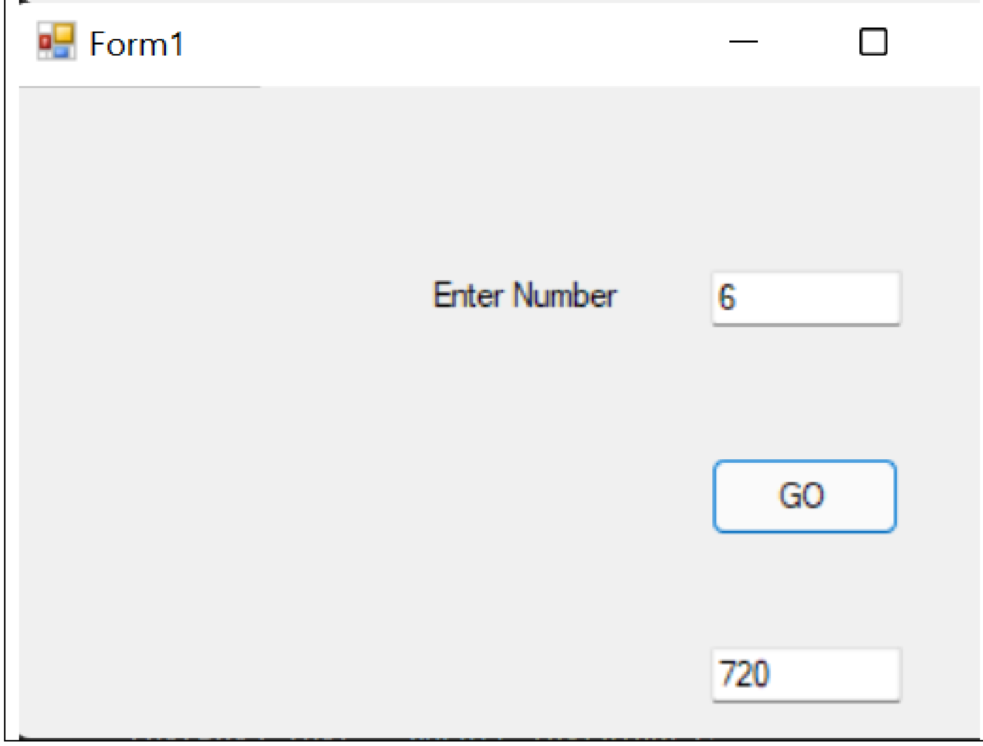
```
using System;
using System.Windows.Forms;
using MathematicsLibrary;

namespace MyApp
{
    public partial class Form1: Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int n = Convert.ToInt32(textBox1.Text);
            int result = Algebra.Factorial(n);
            textBox2.Text = result.ToString();
        }
    }
}
```

OUTPUT:



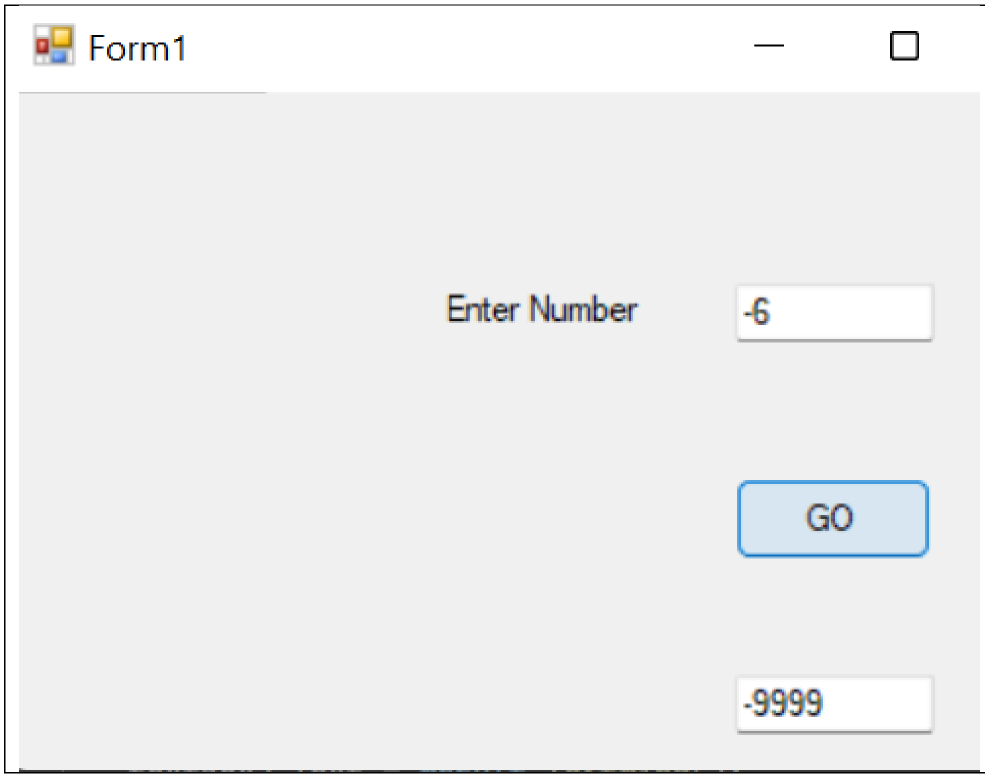
Enter Number 9

-999



Enter Number 6

720



Q7). For the above method, Implement TDD and write 4 test cases and put the code in a word document. put the screen shot of all test cases failing make the test cases pass. put the screen shot.

CODE:

ALGEBRA TEST CLASS:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MathematicsLibrary;
namespace MathematicsLibrary.Tests
{
    [TestClass()]
    public class AlgebraTests
    {
        [TestMethod()]
        public void FactorialTest_Zero_Input()
        {
            //Arrange
            int n = 0;
            int expected = 1;

            //Act
            int actual = Algebra.Factorial(n);

            //Assert
            Assert.AreEqual(expected, actual);
```

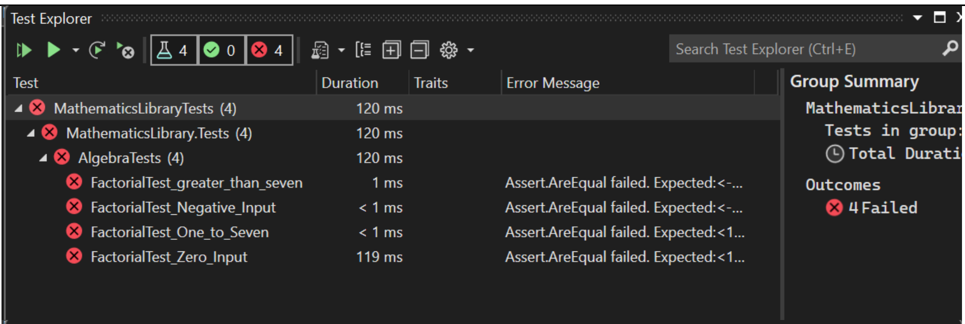
```
}


[TestMethod()]
public void FactorialTest_Negative_Input()
{
   //Arrange
   int n = -1;
   int expected = -9999;

   //Act
   int actual = Algebra.Factorial(n);

   //Assert
   Assert.AreEqual(expected, actual);
}

[TestMethod()]
public void FactorialTest_greater_than_seven()
{
   //Arrange
   int n = 8;
   int expected = -999;

   //Act
   int actual = Algebra.Factorial(n);

   //Assert
   Assert.AreEqual(expected, actual);
}

[TestMethod()]
public void FactorialTest_One_To_Seven()
{
   //Arrange
   int n = 5;
   int expected = 120;

   //Act
   int actual = Algebra.Factorial(n);

   //Assert

   Assert.AreEqual(expected, actual);
```

```
        }
    }
}
```

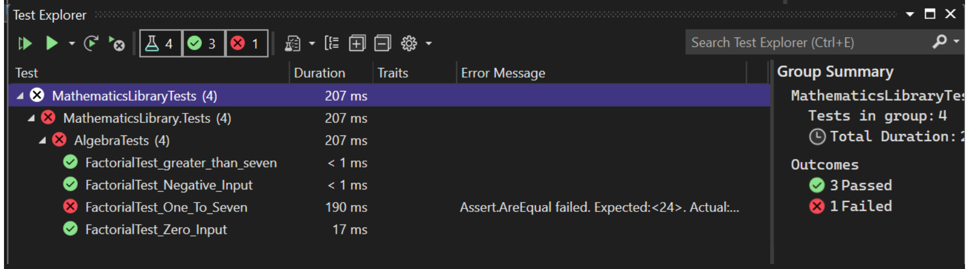
ALGEBRA CLASS:

```
using System;
using System.Collections.Generic;

namespace MathematicsLibrary
{
    /// <summary>
    /// DONE BY: PAVAN
    /// PURPOSE: CREATING AN ALGEBRA CLASS;
    /// </summary>
    public class Algebra
     {
        public static int Factorial(int n)
        {
            if (n == 0)
                return 1;
            else if (n < 0)
                return -9999;
            else if (n > 7)
                return -999;
            else
            {
                int fact = 1;
                for (int i = 0; i < +n; i++)
                    fact = fact * i;
                return fact;
            }
        }
    }
}
```

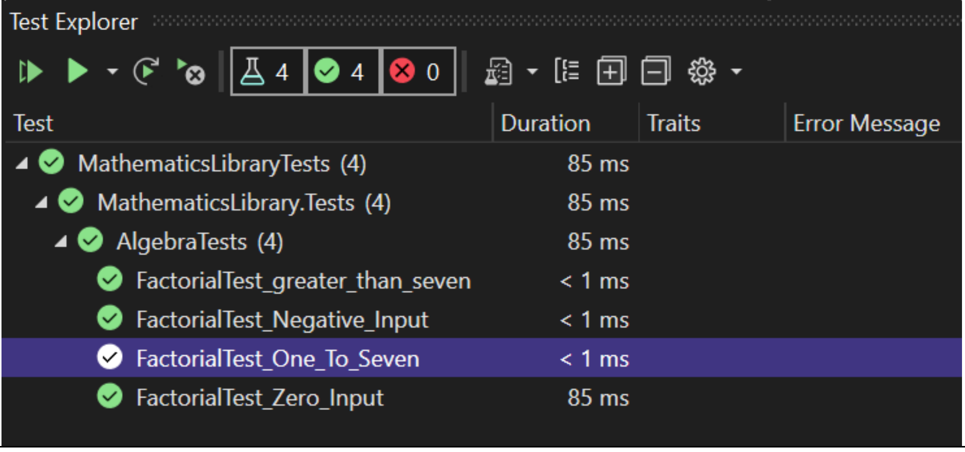
OUTPUT:



```
Test Explorer
4  0  4                                   Search Test Explorer (Ctrl+E)
Test                                   Duration  Traits  Error Message                      Group Summary
MathematicsLibraryTests (4)            120 ms                                               MathematicsLibrar
  MathematicsLibrary.Tests (4)         120 ms                                                 Tests in group:
    AlgebraTests (4)                   120 ms                                                 Total Durati
      FactorialTest_greater_than_seven   1 ms            Assert.AreEqual failed. Expected:<-...   Outcomes
      FactorialTest_Negative_Input     < 1 ms            Assert.AreEqual failed. Expected:<-...     4 Failed
      FactorialTest_One_to_Seven       < 1 ms            Assert.AreEqual failed. Expected:<1...
      FactorialTest_Zero_Input         119 ms            Assert.AreEqual failed. Expected:<1...
```



```
Test Explorer
4  3  1                                   Search Test Explorer (Ctrl+E)
Test                                   Duration  Traits  Error Message                      Group Summary
MathematicsLibraryTests (4)            207 ms                                               MathematicsLibraryTe
  MathematicsLibrary.Tests (4)         207 ms                                                 Tests in group: 4
    AlgebraTests (4)                   207 ms                                                 Total Duration:
      FactorialTest_greater_than_seven < 1 ms                                               Outcomes
      FactorialTest_Negative_Input     < 1 ms                                                 3 Passed
      FactorialTest_One_To_Seven       190 ms            Assert.AreEqual failed. Expected:<24>. Actual:...   1 Failed
      FactorialTest_Zero_Input          17 ms
```



```
Test Explorer
4  4  0
Test                                   Duration  Traits  Error Message
MathematicsLibraryTests (4)             85 ms
  MathematicsLibrary.Tests (4)          85 ms
    AlgebraTests (4)                    85 ms
      FactorialTest_greater_than_seven < 1 ms
      FactorialTest_Negative_Input     < 1 ms
      FactorialTest_One_To_Seven       < 1 ms
      FactorialTest_Zero_Input          85 ms
```

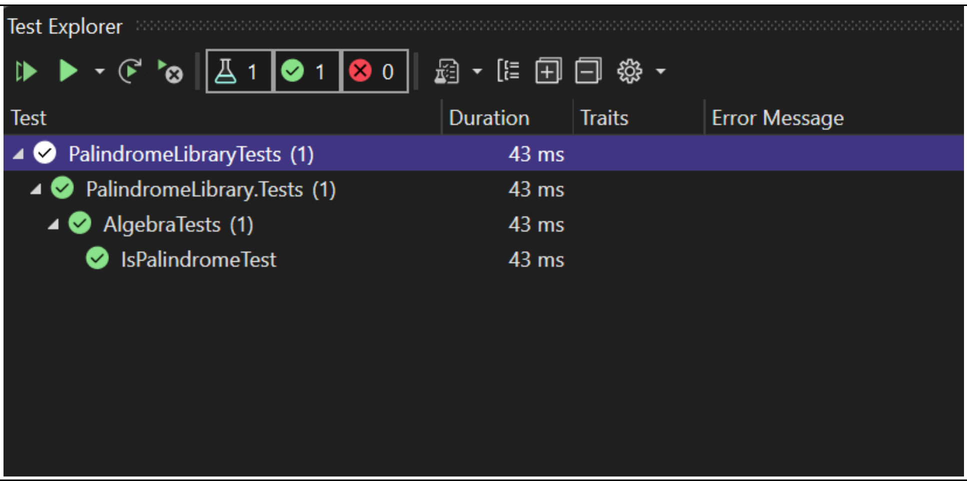
Q8). PALINDROME OR NOT

CODE:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using PalindromeLibrary;
using System;
u
namespace PalindromeLibrary.Tests
{
    [TestClass()]
    public class AlgebraTests
    {
        [TestMethod()]
        public void IsPalindromeTest()
        {
            //Arrange
            int n = 131;
            bool expected = true;

            //Act
           bool actual = Algebra.IsPalindrome(n);
```

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

OUTPUT:



```
Test Explorer
1   1   0
Test                                Duration   Traits   Error Message
◢ PalindromeLibraryTests (1)          43 ms
  ◢ PalindromeLibrary.Tests (1)       43 ms
    ◢ AlgebraTests (1)                43 ms
        IsPalindromeTest              43 ms
```