Day 18 Assignment (16-02-2022)

-By
Ram Charan

1. What is the use of XML

- XML is known as eXtensible Markup Language

- XML is used for universal data transfer mechanism to send data across different platforms.

2. Write the points discussed about xml in the class

- XML is user-defined tags.
- XML has only one root tag.
- XML is used for universal data transfer mechanism to send data across different platforms.
- XML stands for eXtensible Mark-up Language.
- XML is case sensitive.
- Types:
    1.Tag based XML
    2.Attribute based XML

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

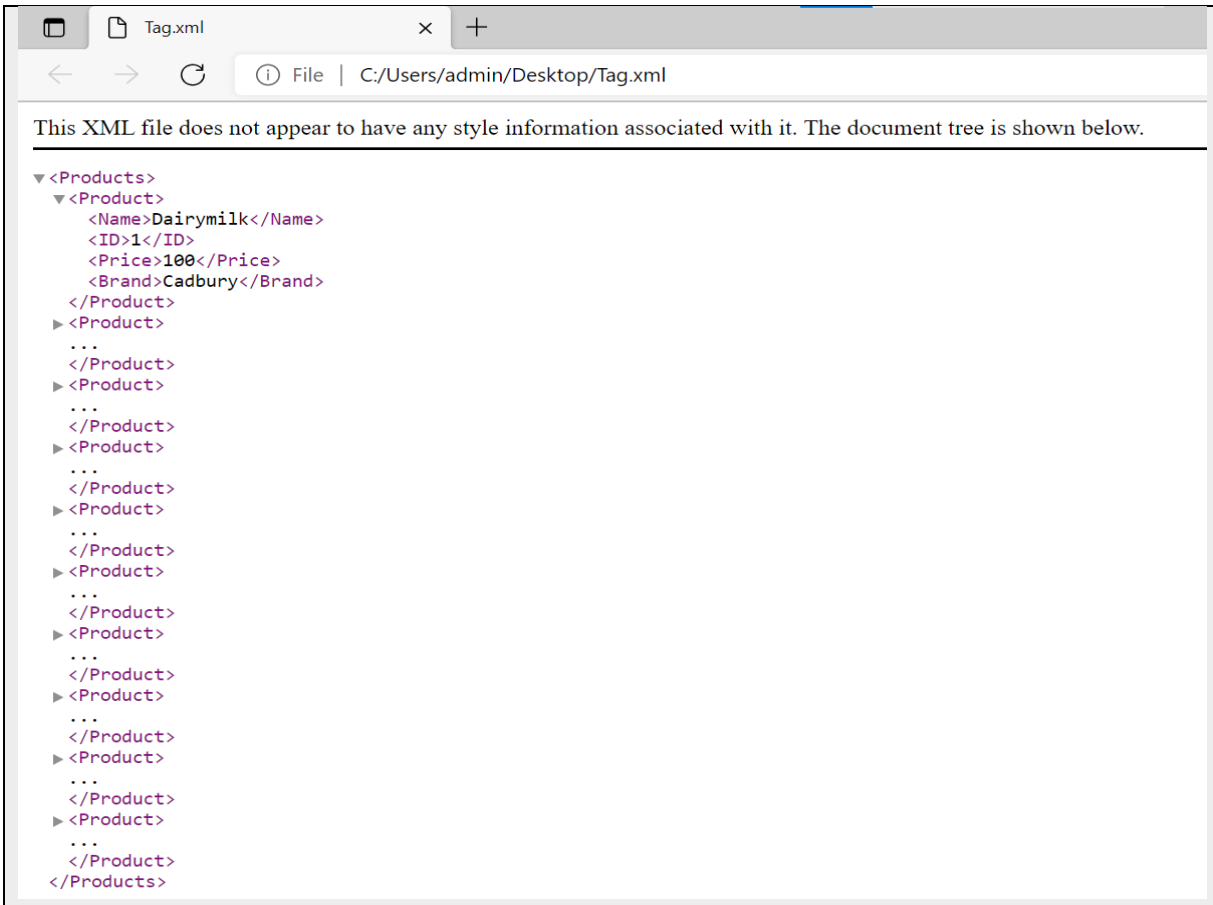
a.) Tag based XML

```
<Products>
 <Product>
  <Name>Dairymilk</Name>
  <ID>1</ID>
  <Price>100</Price>
  <Brand>Cadbury</Brand>
 </Product>
 <Product>
  <Name>5Star</Name>
```

```
 <ID>2</ID>
 <Price>40</Price>
 <Brand>Cadbury</Brand>
</Product>
<Product>
 <Name>Snickers</Name>
 <ID>3</ID>
 <Price>50</Price>
 <Brand>Cadbury</Brand>
</Product>
<Product>
 <Name>Kit-Kat</Name>
 <ID>4</ID>
 <Price>25</Price>
 <Brand>Cadbury</Brand>
</Product>
<Product>
 <Name>Milkybar</Name>
 <ID>5</ID>
 <Price>20</Price>
 <Brand>Cadbury</Brand>
</Product>
<Product>
 <Name>BarOne</Name>
 <ID>6</ID>
 <Price>10</Price>
 <Brand>Cadbury</Brand>
</Product>
<Product>
 <Name>Tresemme</Name>
 <ID>7</ID>
 <Price>5</Price>
 <Brand>Johnson-n-Johnson</Brand>
```

```
  </Product>
<Product>
   <Name>Meera</Name>
   <ID>8</ID>
   <Price>3</Price>
   <Brand>Johnson-n-Johnson</Brand>
  </Product>
<Product>
   <Name>Loreal</Name>
   <ID>9</ID>
   <Price>50</Price>
   <Brand>Loreal</Brand>
  </Product>
<Product>
   <Name>Matrix</Name>
   <ID>10</ID>
   <Price>400</Price>
   <Brand>Matrix</Brand>
  </Product>
</Products>
```

Output:



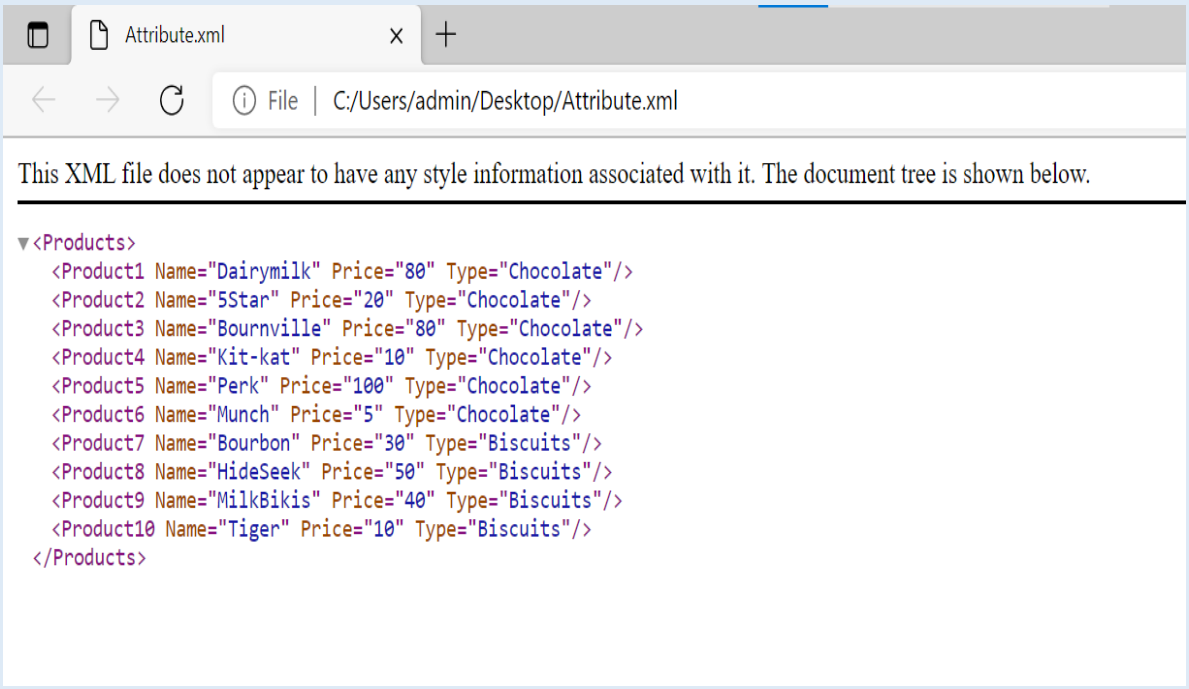
b.) Attribute based XML

Code:

```
<Products>
   <Product1 Name="Dairymilk" Price="80" Type="Chocolate" />
   <Product2 Name="5Star" Price="20" Type="Chocolate" />
   <Product3 Name="Bournville" Price="80" Type="Chocolate" />
   <Product4 Name="Kit-kat" Price="10" Type="Chocolate" />
   <Product5 Name="Perk" Price="100" Type="Chocolate" />
   <Product6 Name="Munch" Price="5" Type="Chocolate" />
   <Product7 Name="Bourbon" Price="30" Type="Biscuits" />
```

```
    <Product8 Name="HideSeek" Price="50" Type="Biscuits"/>
    <Product9 Name="MilkBikis" Price="40" Type="Biscuits" />
    <Product10 Name="Tiger" Price="10" Type="Biscuits" />
</Products>
```

Output:



4. Convert the above xml to JSON and display the JSON data

a.) Attribute xml to JSON data

Output:

```
{
  "Products": {
```

```
"Product1": {
 "-Name": "Dairymilk",
 "-Price": "80",
 "-Type": "Chocolate",
 "-self-closing": "true"
},
"Product2": {
 "-Name": "5Star",
 "-Price": "20",
 "-Type": "Chocolate",
 "-self-closing": "true"
},
"Product3": {
 "-Name": "Bournville",
 "-Price": "80",
 "-Type": "Chocolate",
 "-self-closing": "true"
},
"Product4": {
 "-Name": "Kit-kat",
 "-Price": "10",
 "-Type": "Chocolate",
 "-self-closing": "true"
},
"Product5": {
 "-Name": "Perk",
 "-Price": "100",
 "-Type": "Chocolate",
 "-self-closing": "true"
},
"Product6": {
 "-Name": "Munch",
 "-Price": "5",
```

```
    "-Type": "Chocolate",
    "-self-closing": "true"
   },
   "Product7": {
    "-Name": "Bourbon",
    "-Price": "30",
    "-Type": "Biscuits",
    "-self-closing": "true"
   },
   "Product8": {
    "-Name": "HideSeek",
    "-Price": "50",
    "-Type": "Biscuits",
    "-self-closing": "true"
   },
   "Product9": {
    "-Name": "MilkBikis",
    "-Price": "40",
    "-Type": "Biscuits",
    "-self-closing": "true"
   },
   "Product10": {
    "-Name": "Tiger",
    "-Price": "10",
    "-Type": "Biscuits",
    "-self-closing": "true"
   }
  },
  "#omit-xml-declaration": "yes"
}
```


b. Tag xml to JSON data

Output:

```
{
  "Products": {
   "Product": [
    {
     "Name": "Dairymilk",
     "ID": "1",
     "Price": "100",
     "Brand": "Cadbury"
    },
    {
     "Name": "5Star",
     "ID": "2",
     "Price": "40",
     "Brand": "Cadbury"
    },
    {
     "Name": "Snickers",
     "ID": "3",
     "Price": "50",
     "Brand": "Cadbury"
    },
    {
     "Name": "Kit-Kat",
     "ID": "4",
     "Price": "25",
     "Brand": "Cadbury"
    },
    {
     "Name": "Milkybar",
     "ID": "5",
     "Price": "20",
     "Brand": "Cadbury"
```

```
    },
    {
     "Name": "BarOne",
     "ID": "6",
     "Price": "10",
     "Brand": "Cadbury"
    },
    {
     "Name": "Tresemme",
     "ID": "7",
     "Price": "5",
     "Brand": "Johnson-n-Johnson"
    },
    {
     "Name": "Meera",
     "ID": "8",
     "Price": "3",
     "Brand": "Johnson-n-Johnson"
    },
    {
     "Name": "Loreal",
     "ID": "9",
     "Price": "50",
     "Brand": "Loreal"
    },
    {
     "Name": "Matrix",
     "ID": "10",
     "Price": "400",
     "Brand": "Matrix"
    }
   ]
  },
```

```
  "#omit-xml-declaration": "yes"
}
```

5. Research and write the benefits of JSON over XML (2 or 3 points)

- JSON – JavaScript Object Notation.
- It takes less size (or) memory.
- It is easy to parse.
- It doesnot require or have any tags.

6. For the below requirement, create a layered architecture project with seperate class library for Business logic. create console application create windows(or desktop) application Business Requirement: FIND FACTORIAL OF A NUMBER: 0 = 1 positive number (upto 7) = factorial answer > 7 = -999 (as answer) < 0 = -9999 (as answer) put the screen shots of the output and project (solution explorer) screen shot

Code:

```
---------------------------------------------------------------------------------------------
-------------------------------------

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

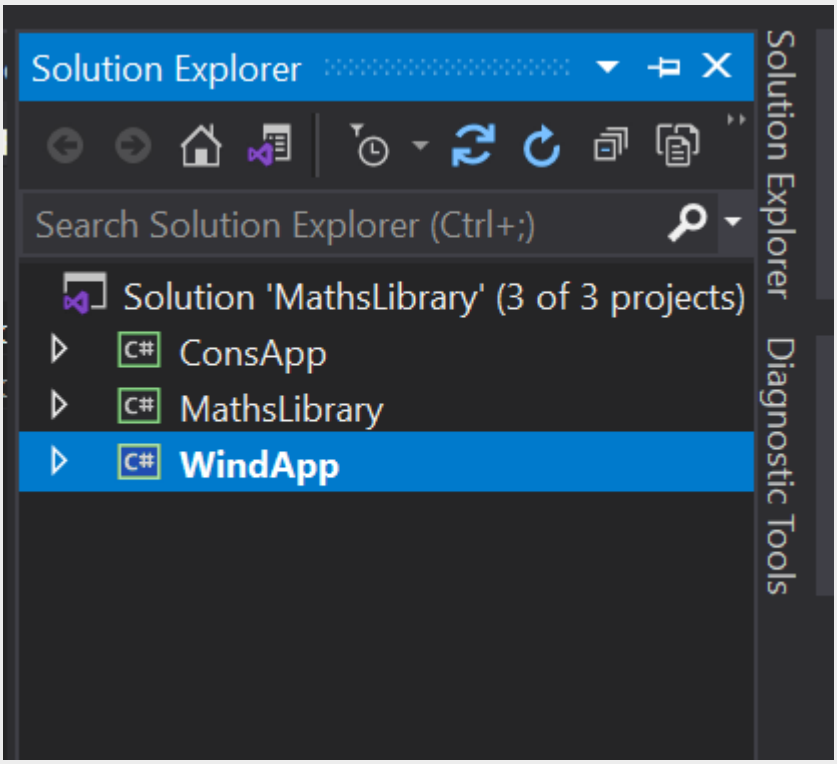
```
using System.Text;
using System.Threading.Tasks;

namespace MathsLibrary
{
    //Author : RC
   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;
      }
   }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using MathsLibrary;


namespace ConsApp
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine(Algebra.Factorial(5));
         Console.ReadLine();
      }
   }
}


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MathsLibrary;

namespace WindApp
{
   public partial class Form1 : Form
   {
      public Form1()
      {
```

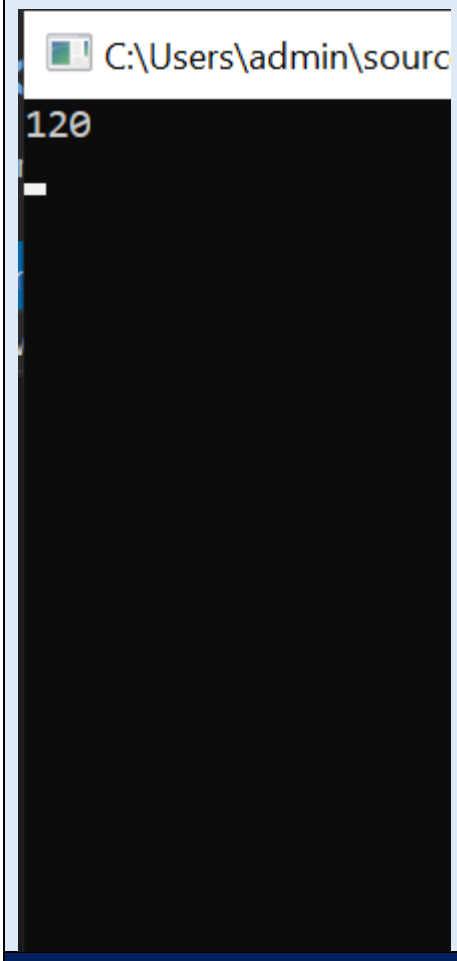
```
        InitializeComponent();
    }

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

Output:

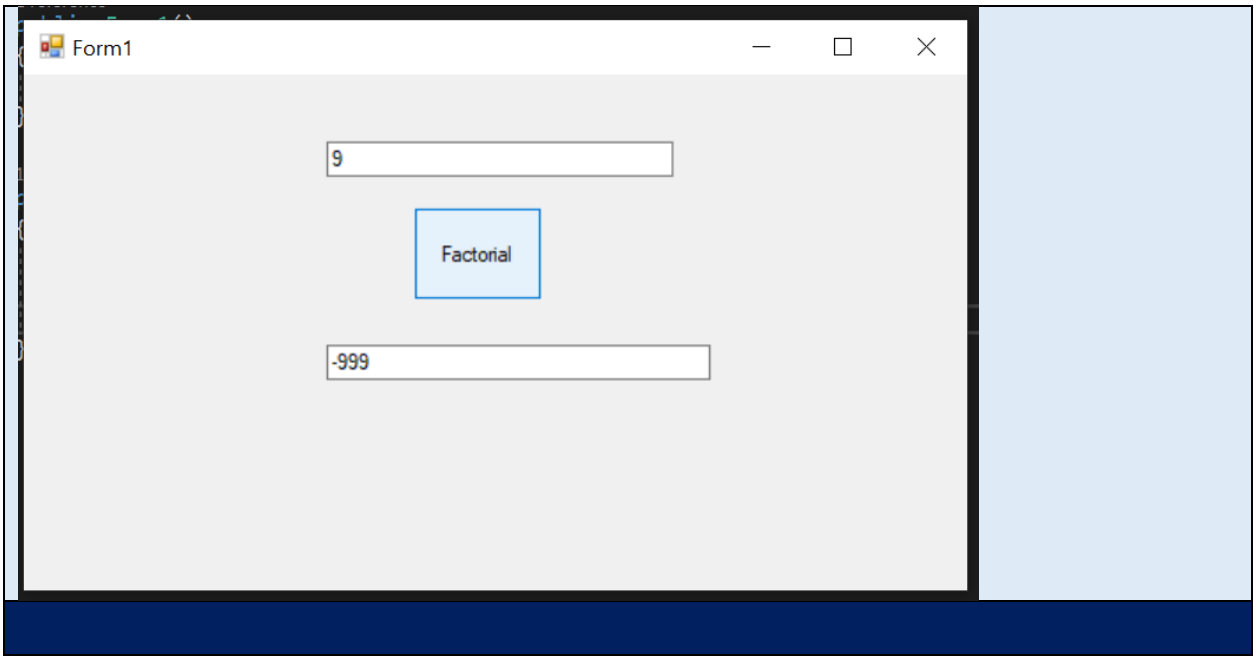
Solution Explorer Screenshot:



Console Application Screenshot:



Windows Application Screenshot:



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

Code:

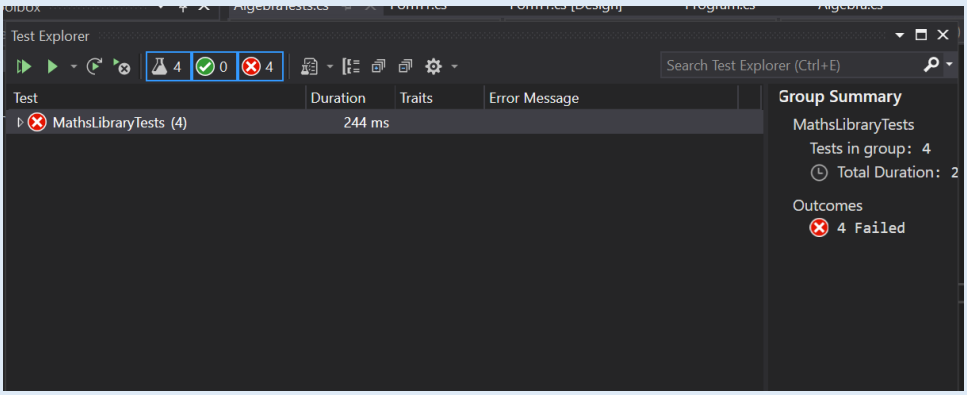
```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MathsLibrary;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathsLibrary.Tests
```

```
{
   //Author :RC
   [TestClass()]
   public class AlgebraTests
   {
      [TestMethod()]
      public void FactorialTest_Zero_Input()
      {
         //Arrange
         int n = 0;
         int expected = 1;

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

         //Assert
         Assert.AreEqual(expected, actual);
      }
      [TestMethod()]
      public void FactorialTest_UptoSeven_Input()
      {
         //Arrange
         int n = 7;
         int expected = 5040;

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

         //Assert
         Assert.AreEqual(expected, actual);
      }
      [TestMethod()]
      public void FactorialTest_LessthanZero_Input()
```

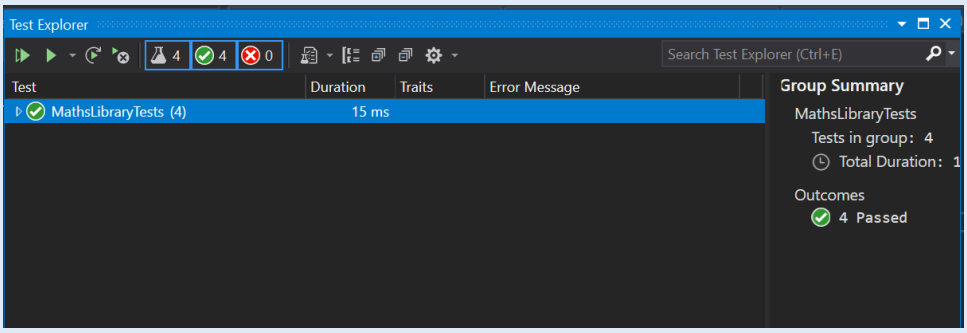
```
    {
        //Arrange
        int n = -5;
        int expected = -9999;

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

        //Assert
        Assert.AreEqual(expected, actual);
    }
    [TestMethod()]
    public void FactorialTest_Greaterthan_Seven_Input()
    {
        //Arrange
        int n = 8;
        int expected = -999;

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

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

  }
}
```

Output:

Failed



Success



8. Add one more method to check if the number is palindrome or not in the above Algebra class and write test case for the same.

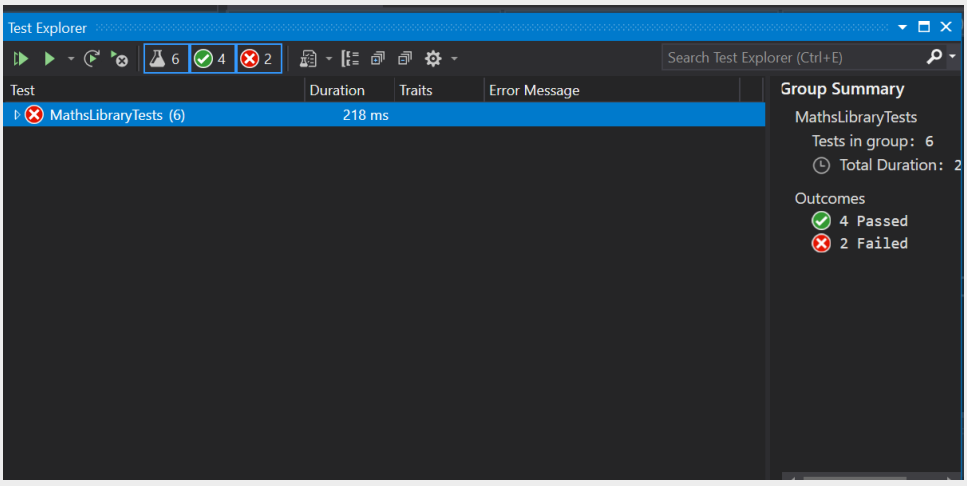
Code:

public static string Palindrome(int n)

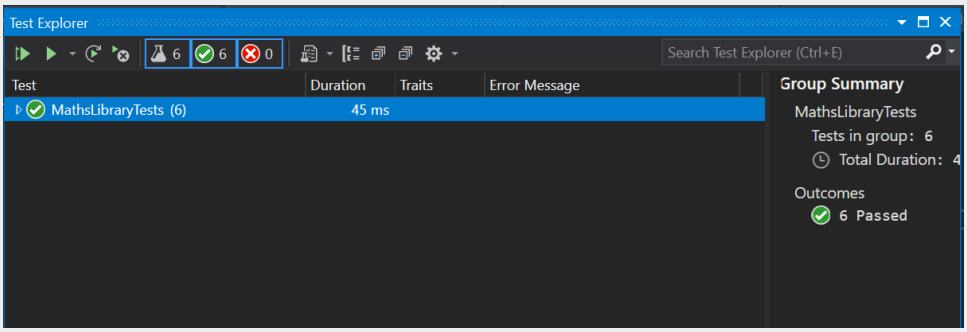
```
    {
        int sum = 0, rem;
        int temp = n;
        while (n > 0)
        {
            rem = n % 10;
            sum = sum * 10 + rem;
            n = n / 10;
        }
        if (temp == sum)
            return "Palindrome";
        else
            return "Not Palindrome";
    }
```

Output:

Failed



Success



End of the Day