|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | DAY18 ASSIGNMENTS  DATE :16/02/2022  DAY :WEDNESDAY  BY  MAVILLA. SAI HARI CHANDANA | | 1. What is the use of XML ?  * “XML” is used for universal data transfer mechanism to send data across different flatforms * “XML” Stands for \_ EXTENSIBLE MARKUP LANGUAGE | | 2.Write the points discussed about XML in the class ? | | * XML is a user defined tags * XML is only 1 root tags * XML is case sensitive * XML is attribute based |  |  | | --- | | 3.create a simple XML to illustrate  a. tag based XML with 10 products  b.attribute based xml | | A.TAG BASED XML.  CODE :  products>  <product1>  <ID>125</ID>  <Name>book</Name>  <price>2000</price>  </product1>  <product2>  <ID>211</ID>  <Name>air conditioner</Name>  < price>3000</price>  </product2>  <product3>  <ID>126</ID>  <Name>laptop</Name>  <price>60000</price>  </product3>  <product4>  <ID>141</ID>  <Name>tv </Name>  <price>60000</price>  </product4>  <product5>  <ID>195</ID>  <Name>refridgerator</Name>  <price>60000</price>  </product4>  <product6>  <ID>114</ID>  <Name>pens</Name>  <price>600</price>  </product6>  <product7>  <ID>156</ID>  <Name>bags</Name>  <price>600</price>  </product7>  <product8>  <ID>110</ID>  <Name>boxes</Name>  <price>600</price>  </product8>  <product9>  <ID>127</ID>  <Name>baskets</Name>  <price>1000</price>  </product9>  <product10>  <ID>106</ID>  <Name>mobile</Name>  <price>25000</price>  </product10>  <products>  OUTPUT : | | B.ATTRIBUTE | | CODE :  <products>  <Product ID ="1" Name ="book" price ="2000" />  <Product ID ="2" Name ="air conditioner " price ="40000" />  <Product ID ="3" Name ="tv" price ="30000" />  <Product ID ="4" Name ="pen" price ="2000" />  <Product ID ="5" Name ="laptop" price ="60000" />  <Product ID ="6" Name ="mobile" price ="12000" />  <Product ID ="7" Name ="book" price ="2000" />  <Product ID ="8" Name ="bags" price ="2000" />  <Product ID ="9" Name ="refridgerator" price ="20000" />  <Product ID ="10" Name ="books" price ="2000" />    </products> |   OUTPUT :     |  | | --- | | 4.convert the above XML to JSON and display the Json data ? | |  | |  | |  |  |  | | --- | | 5.Research and write the benefits of json over XML ? | | * In JSON ,we don’t use tags * Compare with XML, JSON will take less memory * Json is much easier to parse | | 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 | | ALGEBRA :  sing System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  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;  }  }  }   |  | | --- | | CONSOLE APPLICATION | | CODE :  sing System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day18project1  {  public class Program  {  static void Main(string[] args)  {  int n;  Console.WriteLine("Enter number:");  n= Convert.ToInt32(Console.ReadLine());  Console.WriteLine(Algebra.Factorial(n));    Console .ReadLine();  }  }  } | | WINDOWS APPLICATION :  CODE :  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;  namespace WindowsFormsApp2  {  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();  }  }  } | | |

|  |
| --- |
| 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 |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using MathematicsLibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathematicsLibrary.Tests  {  [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\_Zero\_ToSeven\_Input()  {  // Arrange  int n = 3;  int expected = 6;  //Actual  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_LessthanZero\_Input()  {  // Arrange  int n = -6;  int expected = -9999;  //Actual  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_GreaterthanSeven\_Input()  {  // Arrange  int n = 9;  int expected = -999;  //Actual  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
|  |
|  |

|  |
| --- |
|  |
| 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 : |
|  |
|  |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using MathematicsLibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathematicsLibrary.Tests  {  [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\_Zero\_ToSeven\_Input()  {  // Arrange  int n = 3;  int expected = 6;  //Actual  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_LessthanZero\_Input()  {  // Arrange  int n = -6;  int expected = -9999;  //Actual  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_GreaterthanSeven\_Input()  {  // Arrange  int n = 9;  int expected = -999;  //Actual  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void PalindromeTest()  {  //Arrange  int n = 121;  string expected = "Palindrome";  //Actual  string actual = Algebra.Palindrome(n);  //Assert  Assert.AreEqual(expected,actual);  }  [TestMethod()]  public void PalindromeTest\_Wrong\_Input()  {  //Arrange  int n =721 ;  string expected = "Not a Palindrome";  //Actual  string actual = Algebra.Palindrome(n);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
|  |
| Output : |

|  |
| --- |
|  |
|  |