|  |
| --- |
| Day 18(16 Feb) Assignment  by Ramakrishna |

|  |
| --- |
| 1. What is the use of XML |
| * Extensible Markup Language. * XmlDocument tutorial shows how to work with XML in C# with XmlDocument. * Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. XML is often used application configuration, data storage and exchange. |

|  |
| --- |
| 2. Write the points discussed about xml in the class |
| * XML is used for the universal data transfer mechanism to send data across different platform. * XML is case sensitive. * XML Will have user defined tags. * XML should have only one root tags. |

|  |
| --- |
| 3. Create a simple xml to illustrate:  a. Tag based xml with 10 products |
| Code: |
|  |
|  |

|  |
| --- |
| b. Attribute based xml |
|  |

|  |
| --- |
| 4. Convert the above xml to JSON and display the JSON data |
| , |
|  |

|  |
| --- |
| 5. Research and write the benefits of JSON over XML |
| * JSON object has a type whereas XML data is typeless. * JSON does not provide namespace support while XML provides namespaces support. * JSON has no display capabilities whereas XML offers the capability to display data. * JSON is less secured whereas XML is more secure compared to JSON. * JSON supports only UTF-8 encoding whereas XML supports various encoding formats. |

|  |
| --- |
| 6. For the below requirement, create a layered architecture  project with seperate class library for Business logic.    create console application |
| Code: using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Mathametics\_LIbrary  {  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 Mathametics\_LIbrary;  namespace Day\_18\_project\_1  {  internal class Program  {  static void Main(string[] args)  {  Console.WriteLine(Algebra.Factorial(7));  Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| create windows(or desktop) 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;  using Mathametics\_LIbrary;  namespace windows\_app  {  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: |
|  |

|  |
| --- |
| 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. |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Mathametics\_LIbrary  {  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 = 1; i<=n; i++)  fact= fact\*i;  return fact;  }  }  }  }  using Microsoft.VisualStudio.TestTools.UnitTesting;  using Mathametics\_LIbrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Mathametics\_LIbrary.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\_one\_to\_seven\_input()  {  //Arrange  int n = 5;  int expected = 120;  //Act  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_Negative\_input()  {  //Arrange  int n = -4;  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);  }  }  } |
|  |
|  |
|  |
|  |