|  |
| --- |
| Day 18 assignments by lokesh nadella |

|  |
| --- |
| 1. What is the use of xml? |

|  |
| --- |
| XML stands for extensible markup language  It consist of set of codes or tags  The most famous markup language is HTML(hyper text markup language)  XML is a platform independent language |

|  |
| --- |
| 1. Write the points discussed about XML in class? |
| * XML is used for universal data transfer mechanism to send data across different platforms * XML will have user define tags * XML can have only one root tag * XML is case sensitive * There are two types of XML: * Tag based XML * Attribute based XML |

|  |
| --- |
| 1. Create a simple XML to illustrate:   a.tag based XML with 10 products  b.attribute base XML? |
| Code: <products>  <product>  <id>1</id>  <name>pulsar</name>  <price>100000</price>  </product>  <product>  <id>2</id>  <name>ducati</name>  <price>1000000</price>  </product>  <product>  <id>3</id>  <name>tesla</name>  <price>1000000</price>  </product>  <product>  <id>4</id>  <name>yamaha</name>  <price>10000</price>  </product>  <product>  <id>5</id>  <name>r15</name>  <price>100000</price>  </product>  <product>  <id>6</id>  <name>platinum</name>  <price>100000</price>  </product>  <product>  <id>7</id>  <name>passion pro</name>  <price>100000</price>  </product>  <product>  <id>8</id>  <name>pulsar 180</name>  <price>100000</price>  </product>  <product>  <id>9</id>  <name>pulsar220</name>  <price>100000</price>  </product>  <product>  <id>10</id>  <name>ninja kawasaki</name>  <price>100000</price>  </product>  </products> |
| Output: |

|  |
| --- |
| Attribute base XML: |
| CODE: |
| Output: |

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

|  |
| --- |
|  |
| <products>  <product id ="1" name="pulsar" price="10000"></product>  <product id ="2" name="yamaha" price="10000"></product>  <product id ="3" name="r15" price="10000"></product>  <product id ="4" name="pulsar150" price="10000"></product>  <product id ="5" name="platinum" price="10000"></product>  <product id ="6" name="pulsar180" price="10000"></product>  <product id ="7" name="ninja kawasaki" price="10000"></product>  <product id ="8" name="bullet" price="10000"></product>  <product id ="9" name="platinum220" price="10000"></product>  <product id ="10" name="luna" price="10000"></product>  </products> |
| Output: [https://www.freeformatter.com/3.7.0.0/img/minus.gif  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "1",  "@name": "pulsar",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "2",  "@name": "yamaha",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "3",  "@name": "r15",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "4",  "@name": "pulsar150",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "5",  "@name": "platinum",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "6",  "@name": "pulsar180",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "7",  "@name": "ninja kawasaki",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "8",  "@name": "bullet",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "9",  "@name": "platinum220",  "@price": "10000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@id": "10",  "@name": "luna",  "@price": "10000"  }  ] |

|  |
| --- |
| 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:  Algebra.cs:  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace mathematicslibrary  {  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;  }  }  }  } |

|  |
| --- |
| Code:  Program.cs  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using mathematicslibrary;  namespace day18\_project1  {  internal 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();  }  }  } |

|  |
| --- |
|  |

|  |
| --- |
| **7. Research and write the benefits of JSON over XML**  **( 2 or 3 points ).** |

|  |
| --- |
| * JSON is simple * JSON is compact * JSON is easy to learn, easy to read, and easy to understand. |
|  |

|  |
| --- |
| 8 **. 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:  Algebra  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace mathematicallibrary  {  public class algrbra  {  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;  }  return 0;  }  }  } |

|  |
| --- |
| Algebra test  Code:  using Microsoft.VisualStudio.TestTools.UnitTesting;  using mathematicallibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathematicLibrary.Tests  {  [TestClass()]  public class AlgebraTests  {  [TestMethod()]  public void FactorialTest\_Zero\_input()  {  //Arrange  int n = 0;  int expected = 1;  //Act  int actual = algrbra.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 = algrbra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_greater\_than\_Seven\_Input()  {  //Arrange  int n = 9;  int expected = -999;  //Act  int actual = algrbra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_Negitive\_Input()  {  //Arrange  int n = -3;  int expected = -9999;  //Act  int actual = algrbra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |



