**1. Write a C# program to demonstrate Method overloading:**

```
using System;
class Program1 {
        public int Add(int a, int b)
        {
                int sum = a + b;
                return sum;
        }
        public int Add(int a, int b, int c)
        {
                int sum = a + b + c;
                return sum;
        }
        public static void Main(String[] args)
        {
                GFG ob = new GFG();
                int sum1 = ob.Add(1, 2);
                Console.WriteLine("sum of the two " + "integer value : " + sum1);
                int sum2 = ob.Add(1, 2, 3);
                Console.WriteLine("sum of the three " + "integer value : " +
sum2);
        }
}
```

**2. Write a C# program to demonstrate Operator Overloading**

```
using System;
namespace Calculator {
class Calculator {
        public int number1 , number2;
        public Calculator(int num1 , int num2)
        {
                number1 = num1;
                number2 = num2;
        }
        public static Calculator operator -(Calculator c1)
        {
                c1.number1 = -c1.number1;
                c1.number2 = -c1.number2;
                return c1;
        }
        public void Print()
        {
                Console.WriteLine ("Number1 = " + number1);
                Console.WriteLine ("Number2 = " + number2);
```

```
        }
}
class Program2
{
        static void Main(String []args)
        {
                Calculator calc = new Calculator(15, -25);
                calc = -calc;
                calc.Print();
        }
}
}
```

**3. Write a C# program to illustrate the concepts of Events and delegates.**

```
// C# program to illustrate the use of Delegates
using System;
namespace GeeksForGeeks {
        class Geeks {
        public delegate void addnum(int a, int b);
        public delegate void subnum(int a, int b);
                public void sum(int a, int b)
                {
                        Console.WriteLine("(100 + 40) = {0}", a + b);
                }
                public void subtract(int a, int b)
                {
                        Console.WriteLine("(100 - 60) = {0}", a - b);
                }
        public static void Main(String []args)
        {
                Geeks obj = new Geeks();
                addnum del_obj1 = new addnum(obj.sum);
                subnum del_obj2 = new subnum(obj.subtract);
                del_obj1(100, 40);
                del_obj2(100, 60);
        }
        }
}
// C# program to illustrate the use of Events
using System;
namespace SampleApp {
  public delegate string MyDel(string str);
  class EventProgram {
    event MyDel MyEvent;
```

```
    public EventProgram() {
      this.MyEvent += new MyDel(this.WelcomeUser);
    }
    public string WelcomeUser(string username) {
      return "Welcome " + username;
    }
    static void Main(string[] args) {
      EventProgram obj1 = new EventProgram();
      string result = obj1.MyEvent("Tutorials Point");
      Console.WriteLine(result);
    }
  }
}
```

**4. Write a C# .NET program with a function to calculate the count of vowels by taking the input string.**

```
using System;
class GFG{
        public static void Main(){
                char[] inputstring = new char[100];
                int i, vowels, x;
                vowels = 0;
                Console.WriteLine("Please enter the length of the string:\n");
                x = int.Parse(Console.ReadLine());
                Console.WriteLine("Enter string:\n");
                for (i = 0; i < x; i++){
                        inputstring[i] = Convert.ToChar(Console.Read());
                }
                for (i = 0; inputstring[i] != '\0'; i++)
                {
                        if (inputstring[i] == 'a' || inputstring[i] == 'e' ||
                                inputstring[i] == 'i' || inputstring[i] == 'o' ||
                                inputstring[i] == 'u' || inputstring[i] == 'A' ||
                                inputstring[i] == 'E' || inputstring[i] == 'I' ||
                                inputstring[i] == 'O' || inputstring[i] == 'U')
                        {
                                vowels++;
                        }
                }
                Console.WriteLine("\ncount of vowel = " + vowels);
        }
}
```

5. **Write a C# program to demonstrate Generic class.**

```
using System;
public class GFG<T> {
    private T data;
    public T value{
            get{
                    return this.data;
            }
            set{
                    this.data = value;
            }
    }
}
class Test {
    static void Main(string[] args){
            GFG<string> name = new GFG<string>();
            name.value = "GeeksforGeeks";
            GFG<float> version = new GFG<float>();
            version.value = 5.0F;
            Console.WriteLine(name.value);
            Console.WriteLine(version.value);
    }
}
```

6. Write a C# program to demonstrate Generic method

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

namespace GenericMethodAppl {
  class Program {
    static void Swap<T>(ref T lhs, ref T rhs) {
      T temp;
      temp = lhs;
      lhs = rhs;
      rhs = temp;
    }
    static void Main(string[] args) {
      int a, b;
      char c, d;
      a = 10;
```

```
        b = 20;
        c = 'I';
        d = 'V';
        Console.WriteLine("Int values before calling swap:");
        Console.WriteLine("a = {0}, b = {1}", a, b);
        Console.WriteLine("Char values before calling swap:");
        Console.WriteLine("c = {0}, d = {1}", c, d);
        Swap<int>(ref a, ref b);
        Swap<char>(ref c, ref d);
        Console.WriteLine("Int values after calling swap:");
        Console.WriteLine("a = {0}, b = {1}", a, b);
        Console.WriteLine("Char values after calling swap:");
        Console.WriteLine("c = {0}, d = {1}", c, d);
        Console.ReadKey();
    }
  }
}
```

7. Develop a WPF application that demonstrates styles in WPF.

```
<Window x:Class = "XAMLStyle.MainWindow"
    xmlns =
  "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d = "http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc = "http://schemas.openxmlformats.org/markup-
  compatibility/2006"
  xmlns:local = "clr-namespace:XAMLStyle"
  mc:Ignorable = "d" Title = "MainWindow" Height = "350" Width =
  "604">

  <Window.Resources>
    <Style x:Key = "myButtonStyle" TargetType = "Button">
      <Setter Property = "Height" Value = "30" />
      <Setter Property = "Width" Value = "80" />
      <Setter Property = "Foreground" Value = "Blue" />
      <Setter Property = "FontSize" Value = "12" />
      <Setter Property = "Margin" Value = "10" />
    </Style>
  </Window.Resources>

  <StackPanel>
```

```
27.     <Button Content = "Button1" Style = "{StaticResource
    myButtonStyle}" />
28.     <Button Content = "Button2" Style = "{StaticResource
    myButtonStyle}" />
29.     <Button Content = "Button3" Style="{StaticResource
    myButtonStyle}" />
30.   </StackPanel>
31.
32.</Window>
```

8. Develop a WPF application to demonstrates triggers in WPF.

**Property Trigger :**

```
<Window x:Class = "WPFPropertyTriggers.MainWindow"
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  Title = "MainWindow" Height = "350" Width = "604">

  <Window.Resources>
    <Style x:Key = "TriggerStyle" TargetType = "Button">
      <Setter Property = "Foreground" Value = "Blue" />
      <Style.Triggers>
        <Trigger Property = "IsMouseOver" Value = "True">
          <Setter Property = "Foreground" Value = "Green" />
        </Trigger>
      </Style.Triggers>
    </Style>
  </Window.Resources>

  <Grid>
    <Button Width = "100" Height = "70"
      Style = "{StaticResource TriggerStyle}" Content = "Trigger"/>
  </Grid>

</Window>
```

**Data Triggers**

```
<Window x:Class = "WPFDataTrigger.MainWindow"
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  Title = "Data Trigger" Height = "350" Width = "604">
```

```
  <StackPanel HorizontalAlignment = "Center">
    <CheckBox x:Name = "redColorCheckBox"
      Content = "Set red as foreground color" Margin = "20"/>

    <TextBlock Name = "txtblock" VerticalAlignment = "Center"
      Text = "Event Trigger" FontSize = "24" Margin = "20">
      <TextBlock.Style>
        <Style>
          <Style.Triggers>
            <DataTrigger Binding = "{Binding ElementName =
redColorCheckBox, Path = IsChecked}"
              Value = "true">
              <Setter Property = "TextBlock.Foreground" Value = "Red"/>
              <Setter Property = "TextBlock.Cursor" Value = "Hand" />
            </DataTrigger>
          </Style.Triggers>
        </Style>
      </TextBlock.Style>
    </TextBlock>

  </StackPanel>

</Window>
```

**Event Triggers**

```
<Window x:Class = "WPFEventTrigger.MainWindow"
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  Title = "MainWindow" Height = "350" Width = "604">

  <Grid>
    <Button Content = "Click Me" Width = "60" Height = "30">

      <Button.Triggers>
        <EventTrigger RoutedEvent = "Button.Click">
          <EventTrigger.Actions>
            <BeginStoryboard>
              <Storyboard>

                <DoubleAnimationUsingKeyFrames Storyboard.TargetProperty
=
```

```
                    "Width" Duration = "0:0:4">
                    <LinearDoubleKeyFrame Value = "60" KeyTime = "0:0:0"/>
                    <LinearDoubleKeyFrame Value = "120" KeyTime = "0:0:1"/>
                    <LinearDoubleKeyFrame Value = "200" KeyTime = "0:0:2"/>
                    <LinearDoubleKeyFrame Value = "300" KeyTime = "0:0:3"/>
                  </DoubleAnimationUsingKeyFrames>

                  <DoubleAnimationUsingKeyFrames Storyboard.TargetProperty
= "Height"
                    Duration = "0:0:4">
                    <LinearDoubleKeyFrame Value = "30" KeyTime = "0:0:0"/>
                    <LinearDoubleKeyFrame Value = "40" KeyTime = "0:0:1"/>
                    <LinearDoubleKeyFrame Value = "80" KeyTime = "0:0:2"/>
                    <LinearDoubleKeyFrame Value = "150" KeyTime = "0:0:3"/>
                  </DoubleAnimationUsingKeyFrames>

                </Storyboard>
              </BeginStoryboard>
            </EventTrigger.Actions>
          </EventTrigger>
        </Button.Triggers>

      </Button>
    </Grid>

</Window>
```

9. Develop a WPF application to demonstrate data binding in WPF.

   One – Way Data Binding:

```
<Window x:Class = "WPFDataBinding.MainWindow"
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d = "http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc = "http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local = "clr-namespace:WPFDataBinding"
  mc:Ignorable = "d" Title = "MainWindow" Height = "350" Width = "604">

  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height = "Auto" />
      <RowDefinition Height = "Auto" />
      <RowDefinition Height = "*" />
    </Grid.RowDefinitions>
```

```
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width = "Auto" />
      <ColumnDefinition Width = "200" />
    </Grid.ColumnDefinitions>
    <Label Name = "nameLabel" Margin = "2">_Name:</Label>
    <TextBox Name = "nameText" Grid.Column = "1" Margin = "2"
      Text = "{Binding Name, Mode = OneWay}"/>
    <Label Name = "ageLabel" Margin = "2" Grid.Row = "1">_Age:</Label>
    <TextBox Name = "ageText" Grid.Column = "1" Grid.Row = "1" Margin =
"2"
      Text = "{Binding Age, Mode = OneWay}"/>
    <StackPanel Grid.Row = "2" Grid.ColumnSpan = "2">
      <Button Content = "_Show..." Click="Button_Click" />
    </StackPanel>
  </Grid>
</Window>
```

Two – Way Data Binding:

```
<Window x:Class = "WPFDataBinding.MainWindow"
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d = "http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc = "http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local = "clr-namespace:WPFDataBinding"
  mc:Ignorable = "d" Title = "MainWindow" Height = "350" Width = "604">

  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height = "Auto" />
      <RowDefinition Height = "Auto" />
      <RowDefinition Height = "*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width = "Auto" />
      <ColumnDefinition Width = "200" />
    </Grid.ColumnDefinitions>
    <Label Name = "nameLabel" Margin = "2">_Name:</Label>
    <TextBox Name = "nameText" Grid.Column = "1" Margin = "2"
      Text = "{Binding Name, Mode = TwoWay}"/>
    <Label Name = "ageLabel" Margin = "2" Grid.Row = "1">_Age:</Label>
    <TextBox Name = "ageText" Grid.Column = "1" Grid.Row = "1" Margin =
"2"
      Text = "{Binding Age, Mode = TwoWay}"/>
```

```
    <StackPanel Grid.Row = "2" Grid.ColumnSpan = "2">
      <Button Content = "_Show..." Click = "Button_Click" />
    </StackPanel>

  </Grid>
</Window>
```

Main Source code for both one- way and two – way data binding code:

```
using System.Windows;
namespace WPFDataBinding {
  public partial class MainWindow : Window {
    Person person = new Person { Name = "Salman", Age = 26 };
    public MainWindow() {
      InitializeComponent();
      this.DataContext = person;
    }
    private void Button_Click(object sender, RoutedEventArgs e) {
      string message = person.Name + " is " + person.Age;
      MessageBox.Show(message);
    }
  }
  public class Person {
    private string nameValue;
    public string Name {
      get { return nameValue; }
      set { nameValue = value; }
    }
    private double ageValue;
    public double Age {
      get { return ageValue; }
      set {
        if (value != ageValue) {
          ageValue = value;
        }
      }
    }

  }
}
```

10. Write a program to demonstrate StreamReader and StreamWriter.

**StreamReader:**

```
1.using System;
2.using System.IO;
3.public class StreamReaderExample
4.{
5.   public static void Main(string[] args)
6.   {
7.      FileStream f = new FileStream("e:\\output.txt", FileMode.OpenOrCr
    eate);
8.      StreamReader s = new StreamReader(f);
9.
10.       string line=s.ReadLine();
11.       Console.WriteLine(line);
12.
13.       s.Close();
14.       f.Close();
15.   }
16.}
```

**StreamWriter:**

```
1.  using System;
2.  using System.IO;
3.  public class StreamWriterExample
4.  {
5.     public static void Main(string[] args)
6.     {
7.        FileStream f = new FileStream("e:\\output.txt", FileMode.Create);

8.        StreamWriter s = new StreamWriter(f);
9.
10.       s.WriteLine("hello c#");
11.       s.Close();
12.       f.Close();
13.    Console.WriteLine("File created successfully...");
14.   }
15.}
```

11. Write a .NET code to read data from random file and print that data in console window.

```
using System;
using System.IO;
public class Example
{
   public static void Main(string[] args)
   {
        String line;
       try
       {
           StreamReader sr = new StreamReader("C:\\Sample.txt");
         line = sr.ReadLine();
         while (line != null)
         {
            Console.WriteLine(line);
            line = sr.ReadLine();
         }
         sr.Close();
         Console.ReadLine();
       }
       catch(Exception e)
       {
         Console.WriteLine("Exception: " + e.Message);
       }
       finally
       {
         Console.WriteLine("Executing finally block.");
} }
}
```

12. Write a C# program to demonstrate LINQ to XML.

i. **Open Visual Studio->Go to File-> Select New->Select project**

ii. After selecting the project, a new popup will open. From there we have to select "**Asp.Net Empty Web Application**" give the name as **"LINQtoXML" and** click "**OK**" button

iii. To work with LINQ and XML, we will add one XML file in our application. For that, we will **right-click on the application-> Select Add-> Select New Item**,

iv. After clicking on the new item, a new popup will open in that select XML file from the Data Section → Give a name to the XML File->Click Add button.

**Note: No need to write theory points in exam**

When the file is added, we will open it and add the some record as shown below:

```
<?xml version="1.0" encoding="utf-8" ?>
<Employees>
  <employee>
    <empid>1</empid>
    <empname>Akshay</empname>
    <salary>10000</salary>
    <gender>Female</gender>
  </employee>
  <employee>
    <empid>2</empid>
    <empname>Shalu</empname>
    <salary>20000</salary>
    <gender>Female</gender>
  </employee>
  <employee>
    <empid>3</empid>
    <empname>Akki</empname>
    <salary>30000</salary>
    <gender>Male</gender>
  </employee>
  <employee>
    <empid>4</empid>
    <empname>Sateesh</empname>
    <salary>50000</salary>
    <gender>Male</gender>
  </employee>
  <employee>
    <empid>5</empid>
    <empname>Sushmitha</empname>
    <salary>60000</salary>
```

```
    <gender>Female</gender>
  </employee>
</Employees>
```

Now we will show "**XMLFile.xml**" **data** in our application. For that, we have to right click on the application→Select Add->New Item->Select Web Form->Give name as **Default1.aspx** and click "**OK**" button.

Now open "Default1.aspx" page and write the code like as shown below:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div class="GridviewDiv">
<asp:GridView ID="gvDetails" runat="server">
<HeaderStyle CssClass="headerstyle" />
</asp:GridView>
        </div>
    </form>
</body>
</html
```

Now open the code behind the file and write the following code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml.Linq;
public partial class _Default : System.Web.UI.Page
{
```

```
10.   protected void Page_Load(object sender, EventArgs e)
11.   {
12.      if (!Page.IsPostBack)
13.      {
14.         XElement doc = XElement.Load(Server.MapPath("XMLFile.xml"));
15.         var result = from ed in doc.Descendants("employee")
16.                      where Convert.ToInt32(ed.Element("salary").Value) >= 20000
17.                      select new
18.                      {
19.                         Id = ed.Element("empid").Value,
20.                         Name = ed.Element("empname").Value,
21.                         Salary = ed.Element("salary").Value,
22.                         Gender = ed.Element("gender").Value
23.                      };
24.         gvDetails.DataSource = result;
25.         gvDetails.DataBind();
26.      }
27.   }
28.}
```