1. Write a program to demonstrate the use of abstract class. Create two derived classes **`Blue`** and **`Green`** based on a generic **`Color`** class. **`Color`** class should define a template for a method **`Fill(string colorname)`**. This method should be implemented by the classes **`Blue`** and **`Green`**. The **`Fill`** method should display a message "Fill me up with" and then the color name.

Solution:

```
using System;

abstract class Color
{
   public abstract void Fill(string strColor);
}
class Blue : Color
{
   public override void Fill(string strColor)
   {
      Console.WriteLine("Fill me up with " + strColor);
   }
}
class Green : Color
{
   public override void Fill(string strColor)
   {
      Console.WriteLine("Fill me up with " + strColor);
   }
}

class ColorDemo
{
   static void Main()
   {
      Blue b = new Blue();
      b.Fill("Blue");

      Green g = new Green();
      g.Fill("Green");
   }
}
```

2. Write a program to demonstrate polymorphism and the use of keyword `base`. Declare a class **`AppWindow`** having a virtual method **`CreateWindow()`**, displaying message "Window: drawing Window at **`top`**, **`left`**" where **`top`** and **`left`** are integer variables initialized in the constructor. Derive a class **`ListBox`** from **`AppWindow`** and initialize three parameters in its constructor; **`top`**, **`left`** and a string variable **`listBoxContents`**. Override **`CreateWindow()`** with the message "Writing string to the listbox: **`listBoxContents`**"

and also display the virtual method of the base class. Derive another class **`Button`** from **`AppWindow`** and override the virtual method **`CreateWindow()`** with the message "Drawing a button at **`top`**, **`left`**".

Execute **`CreateWindow()`** method for all the three classes.

Solution:

```
using System;

public class AppWindow
{
   //Protected members
   protected int top;
   protected int left;

   // constructor takes two integers to fix location on the
console

   public AppWindow(int top, int left)
   {
      this.top = top;
      this.left = left;
   }
   // simulates drawing the AppWindow
   public virtual void CreateWindow()
   {
      Console.WriteLine("Window: drawing Window at {0}, {1}",
         top, left);
   }

}
// ListBox derives from AppWindow
public class ListBox : AppWindow
{
   private string listBoxContents;

   // constructor adds a parameter and also call base constructor

   public ListBox( int top, int left, string contents):base(top,
left)
   {
      listBoxContents = contents;
   }

   // Overriding CreateWindow
   public override void CreateWindow()
   {
      base.CreateWindow(); // invoking base method
      Console.WriteLine ("Writing string to the listbox: {0}",
listBoxContents);
```

```
   }

}
// Button derives from AppWindow
public class Button : AppWindow
{
   public Button(int top, int left): base(top, left)
   {
   }

   // Overriding CreateWindow
   public override void CreateWindow( )
   {
      Console.WriteLine("Drawing a button at {0}, {1}\n", top,
left);
   }
}
public class Tester
{
   public static void Main()
   {
      AppWindow win = new AppWindow(-110,-0);
      win.CreateWindow( );
      win = new ListBox(3,4,"This is a list box");
      win.CreateWindow();
      win = new Button(5,6);
      win.CreateWindow();
   }
}
```

3. Write a program to demonstrate the use of an interface. Create an interface **`Calculation`** having a method **`Salary()`** which will be implemented by classes **`Accounts`** and **`HR`**. The implemented method should calculate the salary of the respective object (Accounts / HR) and display the same.

Solution:

```
using System;

public interface Calculation
{
   void Salary();
}
public class Accounts : Calculation
{
   private int basic = 6000;
   public void Salary()
   {
      Console.WriteLine("Salary(basic * 5) = "+basic*5);
   }
}
public class HR : Calculation
{
   private int basic = 4000;
   public void Salary()
   {
      Console.WriteLine("Salary (basic * 2) = "+basic*2);
   }
}
public class InterfaceDemo
{
   public static void Main()
   {
      Accounts objacc = new Accounts();
      Console.WriteLine("Accounts Department");
      objacc.Salary();
      Console.WriteLine();

      HR objhr = new HR();
      Console.WriteLine("HR department");
      objhr.Salary();
   }
}
```

4. Create an integer array with five elements and second an empty array of five elements. Copy the elements of the first array to the second. Display both the arrays as well as the array elements of second array in reversed order.

Solution:

```
using System;

class ArrayTest
{
   public static void Main()
   {
      int[] intArray1 = {2,4,6,8,10};
      int[] intArray2 = new int[5];
      Console.WriteLine("First Array");
      ShowArr(intArray1);
      Console.WriteLine("Second Array");
      ShowArr(intArray2);
      Array.Copy(intArray1,intArray2,5);
      Array.Reverse(intArray2);
      Console.WriteLine("Second array after copying and reversing
first array");
      ShowArr(intArray2);
   }

   public static void ShowArr(Array iArray)
   {
      foreach(int cnt in iArray)
      {
         Console.WriteLine(cnt);
      }
   }
}
```

5. Write a program to create a directory called ‘Testdir’ in C:\ drive. Using the same program, delete the directory created after confirming user’s response.

Solution:

```
using System;
using System.IO;

class DirTest
{
   public static void Main()
   {
      string strans;
      Console.WriteLine(@"creating directory c:\Testdir.....");
      Directory.CreateDirectory(@"C:\Testdir");
      Console.WriteLine("Directory created.......");
      Console.WriteLine(@"Delete directory c:\Testdir (y/n)? :");
      strans=Console.ReadLine();
      if(strans=="y")
      {
         Directory.Delete(@"C:\Testdir");
         Console.WriteLine(@"Directory c:\Testdir deleted");
      }
   }
}
```

6. Write a program to create two integer arrays. One array should contain five elements. The other array can be empty. Write a program to copy the contents of the first array to the second and arrange it in ascending order.

Solution:

```
using System;
class Ascend
{
   public static void Main()
   {
      int[] intArray1={12,34,26,108,10};
      int[] intArray2={0,0,0,0,0};
      Console.WriteLine("The contents of array are as follows
:");
      ShowArr(intArray1);
      Array.Copy(intArray1,intArray2,5);
      Array.Sort(intArray2);
      Console.WriteLine("The contents of arrays after copying and 
arranging are as follows  :");
      Console.WriteLine("Array 1 :");
      ShowArr(intArray1);
      Console.WriteLine("Array 2 :");
      ShowArr(intArray2);
   }

   public static void ShowArr(Array iArray)
   {
      foreach(int cnt in iArray)
      {
         Console.WriteLine(cnt);
      }
   }
}
```