Service
Icomplex.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ServiceModel;
using System.Data;
namespace WcfServiceLibrary1
{
    [ServiceContract]
    public interface Icomplex
    {

        [OperationContract]
        Complex Add(Complex complex1, Complex complex2);
        [OperationContract]
        Complex Subtract(Complex complex1, Complex complex2);

    }
}
```

Complex.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ServiceModel;
using System.Data;
using System.Runtime.Serialization;

namespace WcfServiceLibrary1
{
    [DataContract]
    public class Complex
    {
        private int real;
        private int img;
        [DataMember]
        public int Real
        {
            get { return real; }
            set { real = value; }
        }
        [DataMember]
        public int Img
        {
            get { return img; }
            set { img = value; }
        }
    }
}
```

Complexservice.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WcfServiceLibrary1
{
    public class ComplexService : Icomplex
    {
        public Complex Add(Complex complex1, Complex complex2)
        {
            Complex complex3 = new Complex();
            complex3.Real = complex1.Real + complex2.Real;
            complex3.Img = complex1.Img + complex2.Img;

            return complex3;
        }
        public Complex Subtract(Complex complex1, Complex complex2)
        {
            Complex complex3 = new Complex();
            complex3.Real = complex1.Real - complex2.Real;
            complex3.Img = complex1.Img - complex2.Img;

            return complex3;
        }

    }
}
```

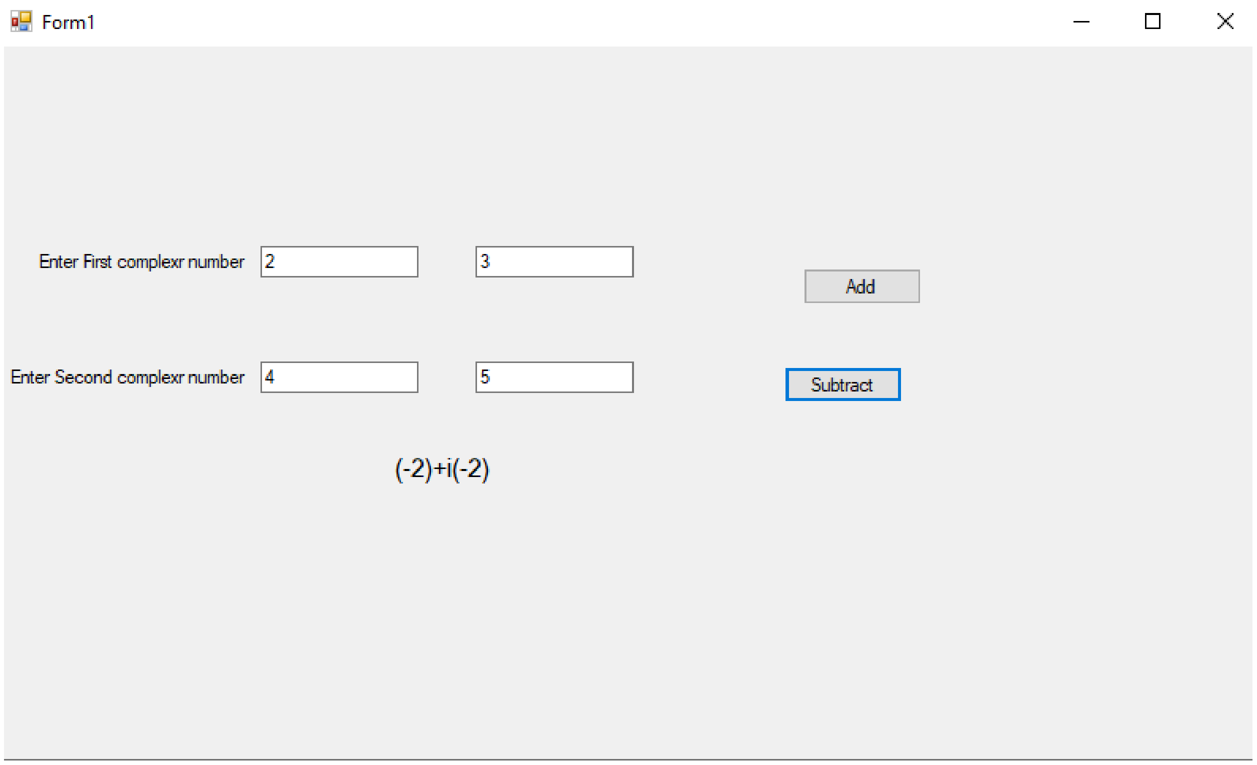
App.config
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" />
  </system.web>
  <!-- When deploying the service library project, the content of the config file must be added to
the host's
  app.config file. System.Configuration does not support config files for libraries. -->
  <system.serviceModel>
    <services>
      <service name="WcfServiceLibrary1.ComplexService">
        <host>
          <baseAddresses>
            <add baseAddress =
"http://localhost:8733/Design_Time_Addresses/WcfServiceLibrary1/Service1/" />
          </baseAddresses>
        </host>
        <!-- Service Endpoints -->
        <!-- Unless fully qualified, address is relative to base address supplied above -->
        <endpoint address="" binding="basicHttpBinding"
contract="WcfServiceLibrary1.Icomplex">
          <!--
              Upon deployment, the following identity element should be removed or replaced to
reflect the
              identity under which the deployed service runs.  If removed, WCF will infer an
appropriate identity
              automatically.
          -->
          <identity>
            <dns value="localhost"/>
          </identity>
        </endpoint>
        <!-- Metadata Endpoints -->
        <!-- The Metadata Exchange endpoint is used by the service to describe itself to clients. -->
        <!-- This endpoint does not use a secure binding and should be secured or removed before
deployment -->
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- To avoid disclosing metadata information,
          set the values below to false before deployment -->
          <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True"/>
          <!-- To receive exception details in faults for debugging purposes,
          set the value below to true.  Set to false before deployment
          to avoid disclosing exception information -->
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>

</configuration>
```

Host
Form1.cs

```
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 System.ServiceModel;
using System.ServiceModel.Description;
using WcfServiceLibrary1;
namespace WCFSERVICEHOST
{
    public partial class Form1 : Form
    {
        ServiceHost sh = null;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            Uri tcpa = new Uri("net.tcp://localhost:8000/TcpBinding");
            sh = new ServiceHost(typeof(ComplexService), tcpa);
            NetNamedPipeBinding pb = new NetNamedPipeBinding();
            NetTcpBinding tcpb = new NetTcpBinding();
            ServiceMetadataBehavior mBehave = new ServiceMetadataBehavior();
            sh.Description.Behaviors.Add(mBehave);
            sh.AddServiceEndpoint(typeof(IMetadataExchange),
MetadataExchangeBindings.CreateMexTcpBinding(), "mex");

            sh.AddServiceEndpoint(typeof(Icomplex), tcpb, tcpa);
            sh.Open();
            label1.Text = "Service Running";
        }
    }
}
```

Client
From1.cs

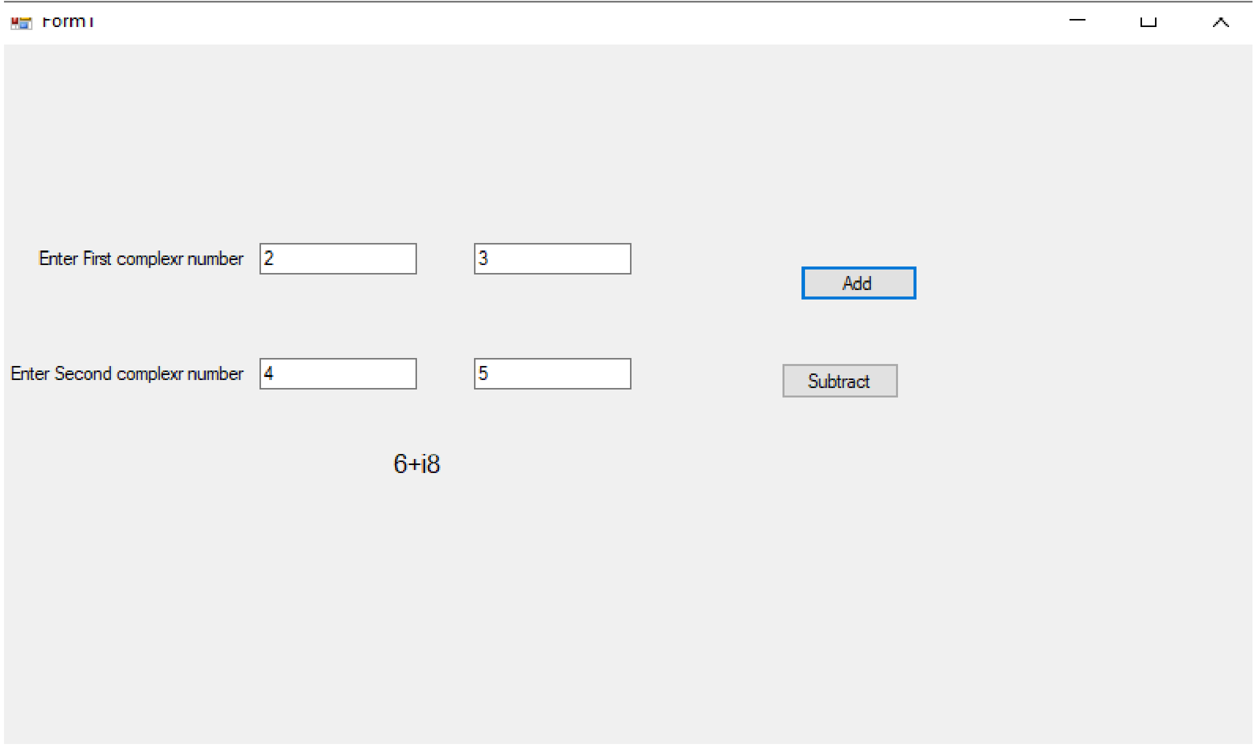
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ComplexClient
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void button1_Click(object sender, EventArgs e)
        {
            TCP.Complex c2 = new TCP.Complex();
            c2.Real = int.Parse(textBox1.Text);
            c2.Img = int.Parse(textBox2.Text);
            TCP.Complex c3 = new TCP.Complex();
            c3.Real = int.Parse(textBox3.Text);
            c3.Img = int.Parse(textBox4.Text);
            TCP.IcomplexClient cilent = new
ComplexClient.TCP.IcomplexClient("NetTcpBinding_Icomplex");
            TCP.Complex c1 = cilent.Add(c2, c3);
            label3.Text = c1.Real.ToString() + "+i" +c1.Img.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            TCP.Complex c2 = new TCP.Complex();
            c2.Real = int.Parse(textBox1.Text);
            c2.Img = int.Parse(textBox2.Text);
            TCP.Complex c3 = new TCP.Complex();
            c3.Real = int.Parse(textBox3.Text);
            c3.Img = int.Parse(textBox4.Text);
            TCP.IcomplexClient cilent = new
ComplexClient.TCP.IcomplexClient("NetTcpBinding_Icomplex");
            TCP.Complex c1 = cilent.Subtract(c2, c3);
            label3.Text = '('+c1.Real.ToString() +')'+ "+i" +'(' +c1.Img.ToString()+')';
        }
    }
}

//Output
```



Enter First complexr number 2 3

Enter Second complexr number 4 5

(-2)+i(-2)



Enter First complexr number 2 3

Enter Second complexr number 4 5

6+i8