In this laboratory project you will build a number of Client/Server applications using C# and the .NET framework. The first will be a simple console application with one-way text transfer. Next, you will modify this application to provide for two-way communication.

One of the simplest types of machine-to-machine communication is client-to-server with a static IP address. In the first part of this laboratory you will implement a client and a server console application using the User Datagram Protocol (UDP). The server will be able to accept a connection from any client, while the client will need the IP address of the server. The source code for the client is shown below.

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace UDPClient
{
    class Class1
    {
        static void Main(string[] args)
        {
            string sendStr = "";
            UdpClient theClient = new UdpClient("server IP address", 9050);
            while (!sendStr.Trim().ToUpper().Equals("END"))
            {
                Console.Write("# ");
                sendStr = Console.ReadLine();
                byte[] myData = new byte[1024];
                myData = Encoding.ASCII.GetBytes(sendStr);
                theClient.Send(myData, myData.Length);
            }
            theClient.Close();
        }
    }
}
```

In this laboratory you may build your own C# applications for the first client and server or you may download the UDPClient.zip and UDPServer.zip project files from the instructor's Web page.

Note that System.Net and System.Net.Sockets have been added to the default list of namespaces provided for the typical console (black-box) application. Review the list of descriptions of objects as you become familiar with the operation of these applications:
sendStr - is the string that will contain the user's message to be sent to the server.

theClient - a UdpClient that connects to the server (server IP address must be included as the first parameter, and a port number. In UDP (and TCP) network communications, labels called ports are used to manage data connections with other computers. A port is an endpoint to a logical connection. The IANA (Internet Assigned Number Authority) has reassigned some port numbers to specific common lines of communication (FTP, DNS, HTTP, etc.)

1. What is the range of Well-Known Port Numbers? ____________ to __________

2. What is the range of Registered Port Numbers? ____________ to __________

myData - is a byte array to hold up to 1024 bytes. The user message is converted to bytes before it is sent to the server.

The source code for UDPServer is shown below:

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace UDPServer
{
    class Class1
    {
        static void Main(string[] args)
        {
            string rcvData = "";
            IPEndPoint IPEP = new IPEndPoint(IPAddress.Any, 9050);
            UdpClient theSock = new UdpClient(IPEP);
            IPEndPoint fromClient;

            while (!rcvData.Trim().ToUpper().Equals("END"))
            {
                byte[] myData = new byte[1024];
                fromClient = new IPEndPoint(IPAddress.Any, 0);
                myData = theSock.Receive(ref fromClient);
                rcvData = Encoding.ASCII.GetString(myData);
                Console.WriteLine(fromClient.ToString() + " " + rcvData);
            }
            theSock.Close();
        }
    }
}
```

IPEP - is an IPEndPoint that will be the other end of the communication connection with the incoming client. Note that the server does not have to know the IP address of the client.

theSock - UdpClient creates a Socket used to send and receive data over a network. Classes deriving from UdpClient can use this property to get or set this Socket.
fromClient - an IPEndPoint that accepts connection from any client. Port = 0 is a well-known port number reserved to receive UDP (or TCP) incoming connections.

recvData - a string to hold the incoming message. The byte array myData is converted to ASCII and placed in this string.

Work with another lab participant to build and test the UDPClient and UDPServer applications.

3. What is special about the string "end" or "END"? ________________________________

4. Can more than one client send to the same server? _______________

5. How could you create two-way communication between two machines using these applications? Hint: you may need to use a second port number. If you try this use 9051 as the other port number.

________________________________________________________________________
________________________________________________________________________

6. In what way could multi-threaded programming be helpful in the implementation of a chat application based on UDP?

________________________________________________________________________
________________________________________________________________________