In this laboratory project we will review the necessary operations for creating a multi-threaded client/server application. This application will implement the concurrent two-way transfer of information between applications using separate client and server threads.

Create a C# console application named TCPClientServer using the code below.

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

namespace TCPClientServer
{
    class Class1
    {
        private static Thread srvThread;
        private static Thread clntThread;
        private static TcpListener theTCPLstn;
        private static TcpClient theTCPSock;
        private static TcpClient theTCPClient;
        private static string remoteIP;

        [STAThread]
        static void Main(string[] args)
        {
            StartTCP();
        }

        private static void StartTCP()
        {
            // call method to create a new server thread
            srvThread = new Thread(new ThreadStart(RunTCPServer));
            srvThread.Start();
            Console.WriteLine("Enter IP Address of remote system... ");
            remoteIP = Console.ReadLine(); // get remote IP as a string

            // call method to create a new client thread
            clntThread = new Thread(new ThreadStart(RunTCPClient));
            clntThread.Start();
        }

        private static void RunTCPServer()
        {
            int received;
            string rcvData = "";
            NetworkStream netStrm;
            theTCPLstn = new TcpListener(IPAddress.Any, 9050);
            theTCPLstn.Start();
            int count = 0;
            while (count < 10)
```csharp
{ try {
    theTCPSock = theTCPLsn.AcceptTcpClient();
    break;
} catch { count += 1; Thread.Sleep(5); }
}
if (count < 10) {
    netStrm = theTCPSock.GetStream();
    while (!rcvData.Trim().ToUpper().Equals("END")) {
        byte[] myData = new byte[1024];
        try {
            received = netStrm.Read(myData, 0, myData.Length);
        } catch { break; }
        if (received != 0) {
            rcvData = Encoding.ASCII.GetString(myData, 0, received);
            Console.WriteLine("Received: "+ rcvData.Trim());
        }
    }
    netStrm.Close();
} else {
    Console.WriteLine("Cannot find client.");
    Console.WriteLine("Closing connections.");
}
try {
    theTCPLsn.Stop();
    theTCPSock.Close();
} catch { }
}
private static void RunTCPClient() {
    string sendStr = "";
    int count = 0;
    while (true) {
        try {
            theTCPClient = new TcpClient(remoteIP, 9050);
            break;
        } catch { count += 1; Thread.Sleep(5); }
    }
    if (count < 10) {
        byte[] myData = new byte[1024];
        // the netStrm is a "text only" stream encoded as ASCII
        NetworkStream netStrm = theTCPClient.GetStream();
        while (!sendStr.Trim().ToUpper().Equals("END")) {
```
Once you have your program running try connecting to another TCPClientServer running on a
different machine in the laboratory.

1. What is the port number for RunTCPClient( ) ?

2. What is the port number for RunTCPserver( ) ?

You may recall that in UDP on server could be connected to multiple clients through the same
incoming port. Also recall that we used a different port for incoming in outgoing messages. This
time we are using the same port for both. The reason for this is that UDP is set up as half-duplex
(i.e. one-way communication) while TCP is almost always full-duplex (two-way communication)

3. Try to connect three TCPClientServer applications to each other and describe the result.

Now go back to a peer-to-peer setup.

4. Can both applications transmit data simultaneously? ______

5. Describe the result of attempting to transmit simultaneously. ___________________________

______________________________________________________________________________
6. When making the initial connection, what happens if the <Enter> key is not pressed on one of the applications?
______________________________________________________________________________
______________________________________________________________________________

7. What happens if the Client is started when the other application is not running? __________
______________________________________________________________________________
______________________________________________________________________________

8. Once two-way communication is established, can either application terminate communication by sending the text "END"? __________ Explain your answer.
______________________________________________________________________________

9. What happens when one of the applications is terminated after communication is established.
______________________________________________________________________________

10. Explain the importance of multi-threading as it is used in this application.
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________