	COLLEGE OF COMPUTING AND INFORMATION SCIENCES		
	Final-Term Assessment Spring 2021 Semester		
Class Id	106383,4,5	**Course Title**	Network programming
Program	BSCS	**Campus / Shift**	North Campus / Morning
Date	29th – April 2021	**Total Marks**	40
Duration	03 hours	**Faculty Name**	Misbah Anwer
Student Id	63583	**Student Name**	AMTA NADEEM

Instructions:

- Filling out Student-ID and Student-Name on exam header is mandatory.
- Do not remove or change any part of exam header or question paper.
- Write down your answers in given space or at the end of exam paper with proper title "Answer for Question# _ _".
- Answers should be formatted correctly (font size, alignment and etc.)
- Handwritten text or image should be on A4 size page with clear visibility of contents.
- Only PDF format is accepted (Student are advise to install necessary software)
- In case of CHEATING, COPIED material or any unfair means would result in negative marking or ZERO.
- A mandatory recorded viva session will be conducted to ascertain the quality of answer scripts were deemed necessary.

- **Caution:** Duration to perform Final-Term Assessment is **03 hours only**. **if you failed to upload answer sheet on LMS (in PDF format) within 03 hours limit, you would be considered as ABSENT/FAILED.**
- Mention complete code with output screenshots

QUESTION NO. 1 **[5+5 Marks]**

a. Explain the following code snippet also explain text in bold from 1 to 5. Rewrite this code for helper classes.

```
IPHostEntry local = Dns.GetHostByName(Dns.GetHostName());
IPEndPoint iep = new IPEndPoint(local.AddressList[0],8000);   (1)
Socket newserver (2) = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
newserver.Bind(iep);(3)
newserver.Listen(5);(4)
Socket newclient(5) = newserver.Accept();
```

A.

IPHostEntry local = Dns.GetHostByName(Dns.GetHostName());

This method can directly pick the IP address. An IPHostEntry instance that contains address information about the host specified in address and GetHostName gets the host name of the local computer.

IPEndPoint iep = new IPEndPoint(local.AddressList[0],**8000**); **(1)**

IP address and port number as parameter

Socket **newserver (2)** = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

Using TCP, that is, Connection Oriented

newserver.Bind(**iep**);**(3)**

This method assign an IP endpoint to a socket instance and is used by server

newserver.Listen**(5);**

This is a method on Server side that contains a parameter backlog indicating how many clients can connect, and always called before accept

Socket **newclient(5)** = newserver.Accept();

This is a method will accept the client and is dependent on clients connect method. When client will request through connect method this accept method will establish a connection

CODE FOR HELPER CLASSES:

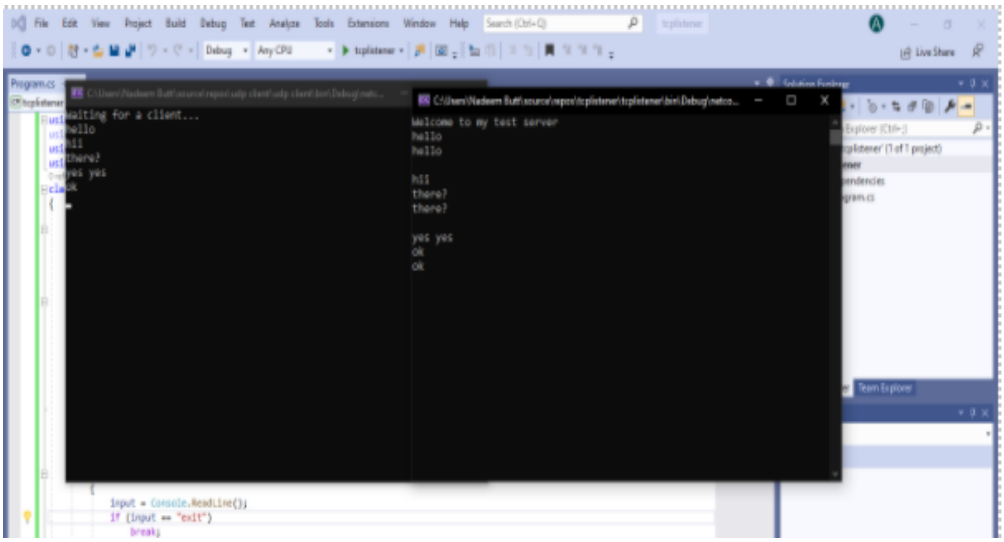
```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpListenerSample {
```

```
    public static void Main()
    { int recv; byte[] data = new byte[1024];
        string input, stringData;
        TcpListener newsock = new TcpListener(9050);
        newsock.Start();
        Console.WriteLine("Waiting for a client...");
        TcpClient client = newsock.AcceptTcpClient();
        NetworkStream ns = client.GetStream();
        string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        ns.Write(data, 0, data.Length);
        while (true) {

            data = new byte[1024];
            recv = ns.Read(data, 0, data.Length);
            if (recv == 0)
                break;
            Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
            ns.Write(data, 0, recv);
            input = Console.ReadLine();
            if (input == "exit")
                break;
            ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
            ns.Flush();
        }
        ns.Close();
        client.Close();
        newsock.Stop();
    }
}
```

CLIENT:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpClientSample
```

```
{
    public static void Main()
    {
        byte[] data = new byte[1024];
        string input, stringData;
        TcpClient server;
        try {
            server = new TcpClient("127.0.0.1", 9050);
        }
        catch (SocketException)
        {
            Console.WriteLine("Unable to connect to server");
            return;
        }
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        while (true)
        {
            input = Console.ReadLine();
            if (input == "exit")
                break;
            ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
            ns.Flush();
            data = new byte[1024];
            recv = ns.Read(data, 0, data.Length);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
        }
        ns.Close();
        server.Close();
    }
}
```



b. Suppose you are tasked to improve the throughput and reliability of real-time video feeds collection from the surveillance cameras inside the city. You incorporate this using a network of intelligent nodes with storage to collect the video feed within an appropriate geographical area. Now, ignoring how cameras storing their feeds on intelligent node, you rather focus on how to collect video feeds from the intelligent nodes in real-time to a central control room datacenter. Create an application using C# network programming for above given scenario, Make suitable assumptions.

In the above situation I will execute TCP on the grounds that here we can see that our work is to get that saved video from nodes to a datacenter. Here we need appropriate and dependable information so we can transfer it to the worker and with regards to reliabilty we generally use TCP as opposed to UDP. I am utilizing a straightforward tcp client and server program utilizing helper classes here is the code:

Datacenter as Server:

```
using System;
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

namespace server
{
   class Program
   {
      static void Main(string[] args)
      {

         IPEndPoint iep = new IPEndPoint(IPAddress.Loopback, 8000);



         int recv;
            byte[] data = new byte[1024];
            TcpListener newsock = new TcpListener(iep);
            newsock.Start();
            Console.WriteLine("Waiting for a intelligent Node...");
            TcpClient client = newsock.AcceptTcpClient();
            NetworkStream ns = client.GetStream();
            string welcome = "Write the  video to upload";
            data = Encoding.ASCII.GetBytes(welcome);
            ns.Write(data, 0, data.Length);
            while (true)
            {
               data = new byte[1024];
```

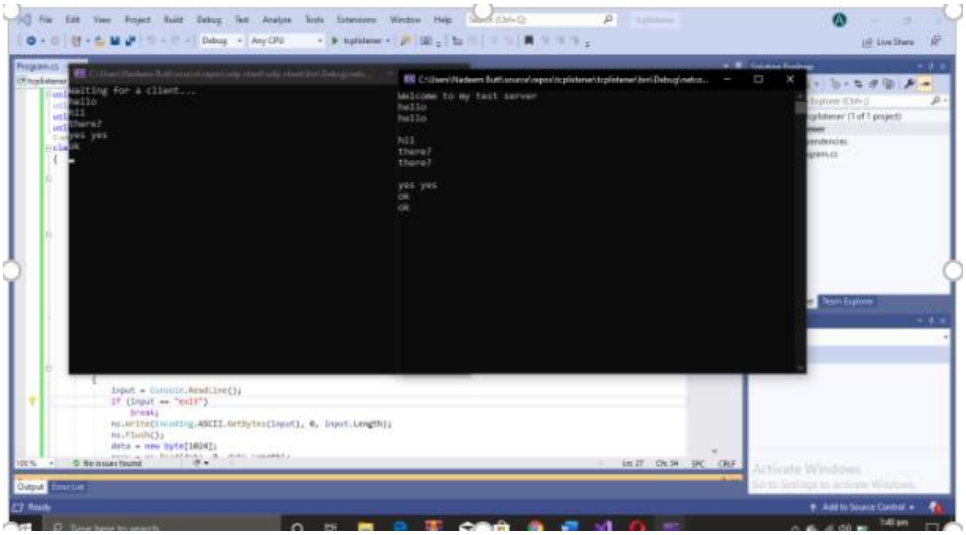
```
                recv = ns.Read(data, 0, data.Length);
                if (recv == 0)
                    break;



                Console.WriteLine(
                Encoding.ASCII.GetString(data, 0, recv));
                ns.Write(data, 0, recv);
            }
            ns.Close();
            client.Close();
            newsock.Stop();
        }
    }
  }
```

Node as a client:

```
using System;
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;


namespace helperr
{
   class Program
   {
      static void Main(string[] args)
```

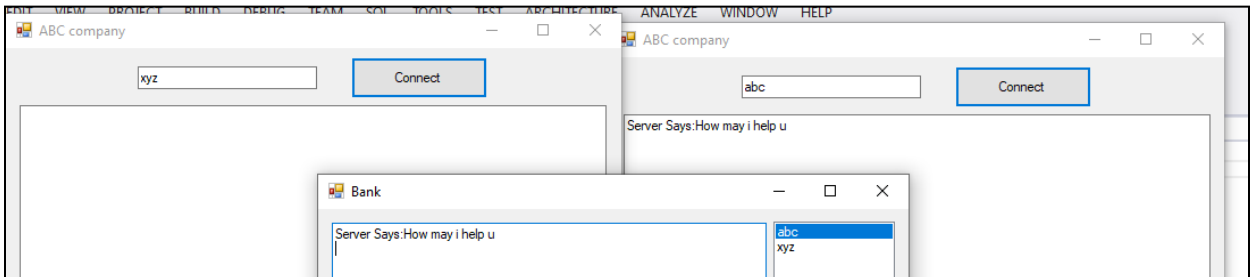
```
{

   byte[] data = new byte[1024];
   string input, stringData;
   TcpClient server;
   try
   {
      server = new TcpClient("127.0.0.1", 8000);
   }
   catch (SocketException)
   {
      Console.WriteLine("Unable to connect to server");
      return;
   }
   NetworkStream ns = server.GetStream();
   int recv = ns.Read(data, 0, data.Length);
   stringData = Encoding.ASCII.GetString(data, 0, recv);
   Console.WriteLine(stringData);
   while (true)
   {
      input = Console.ReadLine();
      if (input == "exit")
         break;
      ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
      ns.Flush();
      data = new byte[1024];
      recv = ns.Read(data, 0, data.Length);
      stringData = Encoding.ASCII.GetString(data, 0, recv);
      Console.WriteLine(stringData);
   }
   Console.WriteLine("Disconnecting from server...");
   ns.Close();
```

```
            server.Close();
        }
    }
}
```



QUESTION NO. 2 **[6+3+3 Marks]**

a. The CEO of organization need to have control on company from anywhere, the organization has three branches, branch one in Islamabad, branch two in Karachi and branch three in Lahore. You need to create asynchronous socket communication using Asynchronous methods.



b. Update part a for group communication among branches.

c. Define blocking, write a C# code that shows blocking also explain how to avoid blocking with the help of code

A

CLIENT;

```
namespace async_client
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      public string name;
      byte[] b = new byte[1024];
      TcpClient client = new
      TcpClient();
      private void button1_Click(object sender, EventArgs e)
      {
         CheckForIllegalCrossThreadCalls = false;
         client.Connect(IPAddress.Loopback,11000);
```

```
            NetworkStream ns =
            client.GetStream(); StreamWriter sw
            = new StreamWriter(ns); name =
            textBox1.Text;
            sw.WriteLine("@name@"+textBox1.
            Text); sw.Flush();
            ns.BeginRead(b,0,b.Length,readmsg,ns);
        }
        private void readmsg(IAsyncResult ar) {

            NetworkStream ns =
            (NetworkStream)ar.AsyncState; int count =
            ns.EndRead(ar);
            richTextBox1.Text += ASCIIEncoding.ASCII.GetString(b, 0,
            count); ns.BeginRead(b,0,b.Length,readmsg,ns);

        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            NetworkStream ns = client.GetStream();
            StreamWriter sw = new
            StreamWriter(ns); sw.WriteLine(name+"
            Says: "+textBox1.Text); sw.Flush();
        }
    }
}
```

Server:

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

        private void Form1_Load(object sender, EventArgs e)
        {
            CheckForIllegalCrossThreadCalls = false;
            TcpListener listener = new TcpListener(IPAddress.Loopback,
            11000); listener.Start(10);
            listener.BeginAcceptTcpClient(new AsyncCallback(client_connect),listener);

        }
        Dictionary<string, TcpClient> lstClients = new Dictionary<string,
        TcpClient>(); byte[] b = new byte[1024];
        private void client_connect(IAsyncResult ar){

            TcpListener listener =
            (TcpListener)ar.AsyncState; TcpClient client
            = listener.EndAcceptTcpClient(ar);
            NetworkStream ns = client.GetStream();
            object [] a = new object[2];
            a[0]=ns;
            a[1]=client;
            ns.BeginRead(b,0,b.Length ,new AsyncCallback(ReadMsg),a);
            listener.BeginAcceptTcpClient(new
            AsyncCallback(client_connect),listener);
```

```
}
private void ReadMsg(IAsyncResult ar){

   object [] a = (object[])ar.AsyncState;
   NetworkStream ns =
   (NetworkStream)a[0]; TcpClient
   client = (TcpClient)a[1];
   int count = ns.EndRead(ar);
   string
   msg=ASCIIEncoding.ASCII.GetString(b,0,count);
   if (msg.Contains("@name@")){

   string name=
   msg.Replace("@name@","");
   lstClients.Add(name, client);
   listBox1.Items.Add(name);

   }
   else{
   richTextBox1.Text+=msg+Environment.NewLi
   ne;

   }
   ns.BeginRead(b,0,b.Length, new AsyncCallback(ReadMsg),a);

}

private void button1_Click(object sender, EventArgs e)
{

      TcpClient client = (TcpClient)lstClients[listBox1.SelectedItem.ToString()];
```

```
            NetworkStream ns =
            client.GetStream(); StreamWriter
            sw = new StreamWriter(ns);

            string texttosend = "Server says : " +
            textBox1.Text; sw.WriteLine(texttosend);

            richTextBox1.Text += texttosend + Environment.NewLine;

            sw.Flush();

        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {

        }
    }
}
```

B

CODE:

SERVER:

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

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Windows.Forms;
using System.IO;

namespace WindowsFormsApp4
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         CheckForIllegalCrossThreadCalls = false;
         TcpListener listener = new TcpListener(IPAddress.Loopback, 11000);
         listener.Start(10);
         listener.BeginAcceptTcpClient(new AsyncCallback(client_connect), listener);

      }
      Dictionary<string, TcpClient> lstClients = new Dictionary<string, TcpClient>();
      byte[] b = new byte[1024];
      private void client_connect(IAsyncResult ar)
      {
```

```
    TcpListener listener = (TcpListener)ar.AsyncState;
    TcpClient client = listener.EndAcceptTcpClient(ar);
    NetworkStream ns = client.GetStream();
    object[] a = new object[2];
    a[0] = ns;
    a[1] = client;
    ns.BeginRead(b, 0, b.Length, new AsyncCallback(ReadMsg), a);
    listener.BeginAcceptTcpClient(new AsyncCallback(client_connect), listener);

}

private void ReadMsg(IAsyncResult ar)
{

    object[] a = (object[])ar.AsyncState;
    NetworkStream ns = (NetworkStream)a[0];
    TcpClient client = (TcpClient)a[1];
    int count = ns.EndRead(ar);
    string msg = ASCIIEncoding.ASCII.GetString(b, 0, count);
    if (msg.Contains("@name@"))
    {

        string name = msg.Replace("@name@", "");
        lstClients.Add(name, client);
        listBox1.Items.Add(name);

    }
    else
    {
        richTextBox1.Text += msg + Environment.NewLine;

    }
```

```
    ns.BeginRead(b, 0, b.Length, new AsyncCallback(ReadMsg), a);

}

private void button1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked == true)
    {
        string items = "";
        foreach (var item in listBox1.Items)
        {
            TcpClient client = (TcpClient)lstClients[item.ToString()];

            NetworkStream ns = client.GetStream();
            StreamWriter sw = new StreamWriter(ns);

            string texttosend = "Server says : " + textBox1.Text;
            sw.WriteLine(texttosend);

            richTextBox1.Text += texttosend + Environment.NewLine;

            sw.Flush();

        }

    }

    else if (radioButton1.Checked == false)
    {

        TcpClient client = (TcpClient)lstClients[listBox1.SelectedItem.ToString()];
```

```
            NetworkStream ns = client.GetStream();
            StreamWriter sw = new StreamWriter(ns);

            string texttosend = "Server says : " + textBox1.Text;
            sw.WriteLine(texttosend);

            richTextBox1.Text += texttosend + Environment.NewLine;

            sw.Flush();

          }
          radioButton1.Checked = false;
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {

        }
    }
}
```

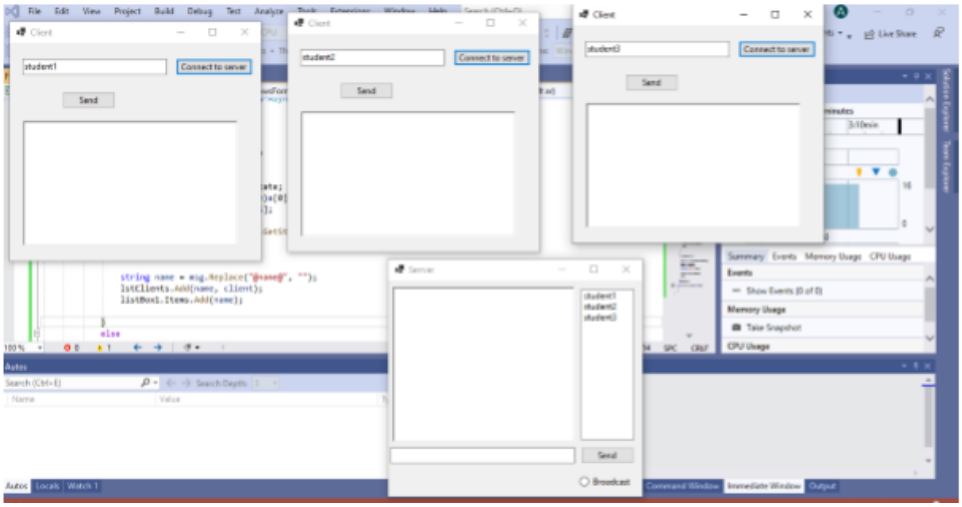
CLIENT CODE:

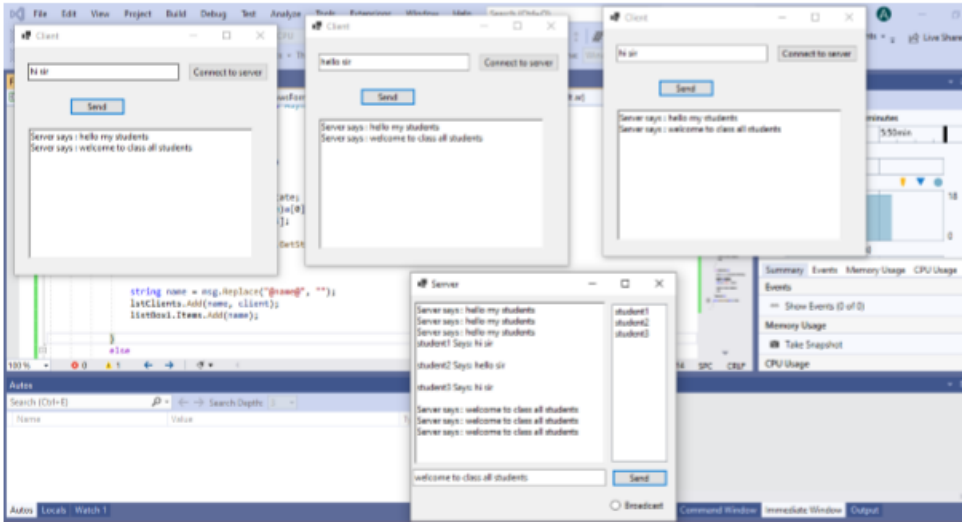
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;



namespace client_ASC
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public string name;
        byte[] b = new byte[1024];
        TcpClient client = new TcpClient();



        private void Form1_Load(object sender, EventArgs e)
        {


        }


        private void button2_Click(object sender, EventArgs e)
        {
            CheckForIllegalCrossThreadCalls = false;
            client.Connect(IPAddress.Loopback, 8001);
```

```
            NetworkStream ns = client.GetStream();
            StreamWriter sw = new StreamWriter(ns);
            name = textBox1.Text;
            sw.WriteLine(textBox1.Text);
            sw.Flush();
            ns.BeginRead(b, 0, b.Length, readmsg, ns);
        }
        private void readmsg(IAsyncResult ar)
        {

            NetworkStream ns = (NetworkStream)ar.AsyncState;
            int count = ns.EndRead(ar);
            richTextBox1.Text += ASCIIEncoding.ASCII.GetString(b, 0, count);
            ns.BeginRead(b, 0, b.Length, readmsg, ns);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            NetworkStream ns = client.GetStream();
            StreamWriter sw = new StreamWriter(ns);
            sw.WriteLine(name + " Says: " + textBox1.Text);
            sw.Flush();

        }
    }
}
```

OUTPUT:





C

Blocking is when a server is waiting for client's request but the client doesnot send request then server will be blocked after waiting for client especially in multi client. The solution is we can use non-blocking sockets, socket multiplexing and aschrysonous socket. Using threads By blocking state equal to false

BLOCKING CODE:

```
TcpListener newserver = new TcpListener(9050);
newserver.Start();
TcpClient newclient = newserver.AcceptTcpClient();
NetworkStream ns = newclient.GetStream();
byte[] outbytes = Encoding.ASCII.GetBytes("Testing");
ns.Write(outbytes, 0, outbytes.Length);
byte[] inbytes = new byte[1024];
ns.Read(inbytes, 0, inbytes.Length);
string instring = Encoding.ASCII.GetString(inbytes);
Console.WriteLine(instring);
ns.Close();
newclient.Close();
newserver.Stop();
```

HOW TO AVOID BLOCKING:

MULTI CLIENT APPLICATION

1) ASYNCHRONOUS METHOD

SERVER CODE

```
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
```

```
using System.IO;


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

private void Form1_Load(object sender, EventArgs e)
{
CheckForIllegalCrossThreadCalls = false;
TcpListener listener = new TcpListener(IPAddress.Loopback, 11000);
listener.Start(10);
listener.BeginAcceptTcpClient(new AsyncCallback(client_connect),listener);

}
Dictionary<string, TcpClient> lstClients = new Dictionary<string, TcpClient>();
byte[] b = new byte[1024];
private void client_connect(IAsyncResult ar){

TcpListener listener = (TcpListener)ar.AsyncState; TcpClient client =
listener.EndAcceptTcpClient(ar); NetworkStream ns = client.GetStream();
object [] a  = new object[2];
a[0]=ns;
a[1]=client;
ns.BeginRead(b,0,b.Length ,new AsyncCallback(ReadMsg),a);
listener.BeginAcceptTcpClient(new AsyncCallback(client_connect),listener);

}
private void ReadMsg(IAsyncResult ar){

object [] a = (object[])ar.AsyncState; NetworkStream ns = (NetworkStream)a[0]; TcpClient client
= (TcpClient)a[1];
```

```
int count = ns.EndRead(ar);
string msg=ASCIIEncoding.ASCII.GetString(b,0,count);
if (msg.Contains("@name@")){

string name= msg.Replace("@name@",""); lstClients.Add(name, client); listBox1.Items.Add(name);

} else{ richTextBox1.Text+=msg+Environment.NewLine;
```

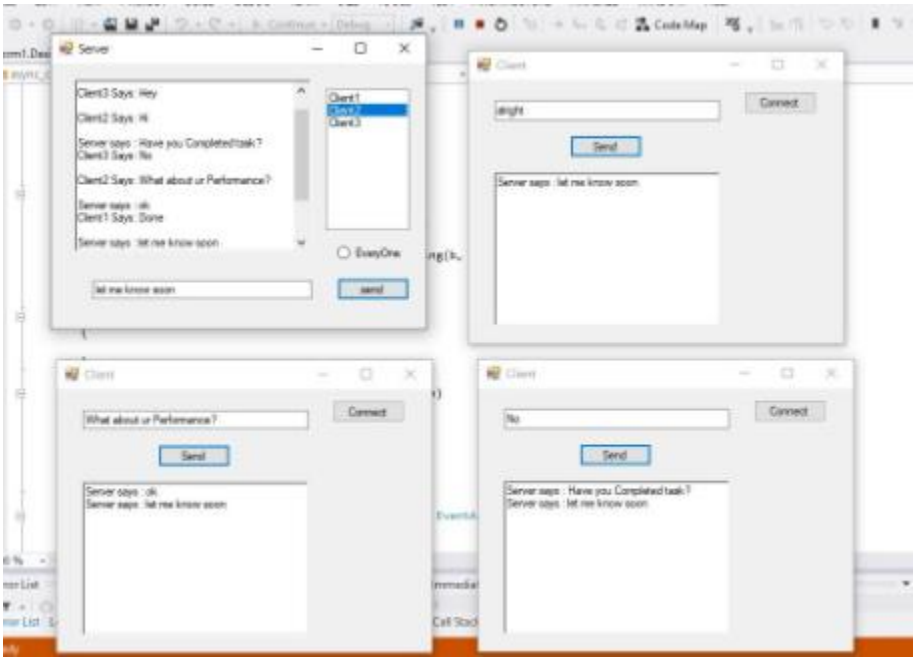
```
ns.BeginRead(b,0,b.Length, new AsyncCallback(ReadMsg),a);


}


private void button1_Click(object sender, EventArgs e)

{


if (radioButton1.Checked == true)

{

string items = "";

foreach (var item in listBox1.Items)

{

//items += item.ToString() + " ";

TcpClient client = (TcpClient)lstClients[item.ToString()];


NetworkStream ns = client.GetStream(); StreamWriter sw = new StreamWriter(ns);


string texttosend = "Server says : " + textBox1.Text;

sw.WriteLine(texttosend);


richTextBox1.Text += texttosend + Environment.NewLine;


sw.Flush();


}


}


else if (radioButton1.Checked == false)

{

TcpClient client = (TcpClient)lstClients[listBox1.SelectedItem.ToString()]; NetworkStream ns =
client.GetStream();

StreamWriter sw = new StreamWriter(ns);


string texttosend = "Server says : " + textBox1.Text;

sw.WriteLine(texttosend);
```

```
richTextBox1.Text += texttosend + Environment.NewLine;


sw.Flush();


}
radioButton1.Checked = false;
}


}
}
```

CLIENT CODE

```
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.IO;

namespace async_client
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
public string name;
byte[] b = new byte[1024];
TcpClient client = new TcpClient();
private void button1_Click(object sender, EventArgs e)
{
CheckForIllegalCrossThreadCalls = false;
client.Connect(IPAddress.Loopback,11000);
```

```
NetworkStream ns = client.GetStream(); StreamWriter sw = new StreamWriter(ns); name =
textBox1.Text; sw.WriteLine("@name@"+textBox1.Text); sw.Flush();
ns.BeginRead(b,0,b.Length,readmsg,ns);

}

private void readmsg(IAsyncResult ar) {


NetworkStream ns = (NetworkStream)ar.AsyncState;

int count = ns.EndRead(ar);

richTextBox1.Text += ASCIIEncoding.ASCII.GetString(b, 0, count);

ns.BeginRead(b,0,b.Length,readmsg,ns);


}

private void button2_Click(object sender, EventArgs e)

{

NetworkStream ns = client.GetStream(); StreamWriter sw = new StreamWriter(ns);
sw.WriteLine(name+" Says: "+textBox1.Text); sw.Flush();

}
```



2) TRADITIONAL METHOD

Server CODE

```
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Threading.Tasks;

namespace server__using_thread
{
class MultiThreadedTcpServer
{
public static void Main(string[] args)
{
int port = 9050;
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

IPEndPoint endpoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), port);
server.Bind(endpoint);

server.Listen(10);
Console.WriteLine("Waiting for clients on port " + port);

while (true)
{
catch (Exception)
{
Console.WriteLine("Connection failed on port " + port);
}
```

```
}

}

}


class ConnectionHandler

{


private Socket client; private NetworkStream ns; private StreamReader reader; private
StreamWriter writer;


private static int connections = 0;


public ConnectionHandler(Socket client)

{

this.client = client;

}


public void HandleConnection()

{

byte[] data = new byte[1024];

int recv;

string input;

ns = new NetworkStream(client); reader = new StreamReader(ns); writer = new StreamWriter(ns);
connections++;

Console.WriteLine("New client accepted: {0} active connections",


string welcome = "Welcome to my test server";

/  writer.WriteLine("Welcome to my server");

writer.Flush();




data = Encoding.ASCII.GetBytes(welcome); client.Send(data, data.Length, SocketFlags.None); while
(true)

{

data = new byte[1024];

//  Console.WriteLine("Server received : ");
```

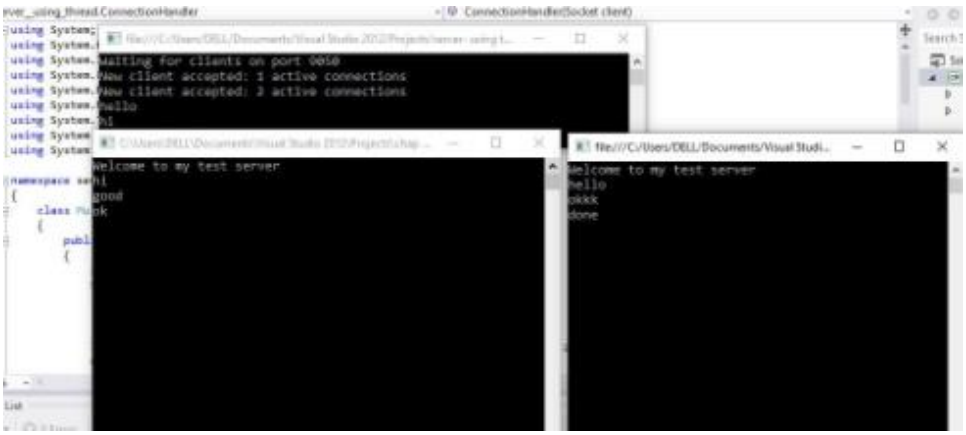
```
recv = client.Receive(data);
if (recv == 0)
break;
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
//   Console.WriteLine("Server sent :");
input = Console.ReadLine();
if (input == "exit")
break;
client.Send(Encoding.ASCII.GetBytes(input));
}

/*
while (true)
{
input = reader.ReadLine();
if (input.Length == 0 || input.ToLower() == "exit")
break;

writer.WriteLine(input);
writer.Flush();
}
```

Client CODE

```
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace chap_5___5._9__Network_Stream_Tcp_Client
{
class Program
{
static void Main(string[] args)
{

byte[] data = new byte[1024];
string input, stringData;
int recv;
```

```
IPEndPoint ipep = new IPEndPoint(
IPAddress.Parse("127.0.0.1"), 9050);
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
try
{
server.Connect(ipep);
}
catch (SocketException e)
{
Console.WriteLine("Unable to connect to server.");
Console.WriteLine(e.ToString());
return;
}
NetworkStream ns = new NetworkStream(server);
if (ns.CanRead)
{
}
else
{




}
recv = ns.Read(data, 0, data.Length);
stringData = Encoding.ASCII.GetString(data, 0, recv); Console.WriteLine(stringData);



Console.WriteLine("Error: Can't read from this socket");
ns.Close(); server.Close(); return;
while (true)
{
input = Console.ReadLine();
if (input == "exit")
```

```
break if (ns.CanWrite)
{

ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
ns.Flush();
}
recv = ns.Read(data, 0, data.Length);
stringData = Encoding.ASCII.GetString(data, 0, recv); Console.WriteLine(stringData);
}
Console.WriteLine("Disconnecting from server..."); Console.ReadKey();
ns.Close(); server.Shutdown(SocketShutdown.Both); server.Close();
}
}
}
```



QUESTION NO. 3 **[4+4 Marks]**

a. How TCP ensure packet reliability? How we can acknowledge the UDP packets? Explain and write code. What are the parameters needed for four synchronous socket methods defined in Dns class?

A

FOR UDP:
1.send data to remote device
2.start with a timer, and set it for a period of time.
3.then wait for the reponse from the remote device and when it arrives stop the timer .
4 and if the timer expires before the response then go back and repeat step 1

TCP is a 3 way hand shaking protocol After the ports, the next fields in the TCP header are the sequence and acknowledgement numbers. These values allow TCP to track packets

and ensure they are received in the proper order from the network. If any packets are missing, the TCP system can request a retransmission of the missing packets and reassemble the data stream before passing it off to the application. By comparing the TCP sequence numbers of each packet, you can determine if packets are being frequently retransmitted (repeated or overlapping sequence numbers in a TCP session)

There are four synchronous methods defined in the Dns class:

GetHostName()

GetHostByName()

GetHostByAddress()

Resolve()

b. Explain the issue in the following code also fix the issue (if any) and rewrite the client code

```
byte[] data = new byte[1024];
string stringData;
IPEndPoint ipep = new IPEndPoint(
IPAddress.Parse("127.0.0.1"), 9050);
Socket server = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
try
{
    server.Connect(ipep);
}
catch (SocketException e)
{
    Console.WriteLine("Unable to connect to server.");
    Console.WriteLine(e.ToString());
    return;
}
int recv = server.Receive(data);
stringData = Encoding.ASCII.GetString(data, 0, recv);
Console.WriteLine(stringData);
server.Send(Encoding.ASCII.GetBytes("message 1"));
server.Send(Encoding.ASCII.GetBytes("message 2"));
server.Send(Encoding.ASCII.GetBytes("message 3"));
server.Send(Encoding.ASCII.GetBytes("message 4"));
server.Send(Encoding.ASCII.GetBytes("message 5"));
Console.WriteLine("Disconnecting from server...");
server.Shutdown(SocketShutdown.Both);
server.Close();
```

B.

This is a badtcp client program the issue in this code is that we are expecting that TCP will send message similarly we are advising it in our program however that is false every time we run this code we will get the yield in various grouping yet we do have an answer for it here is the code for the solutions. I will utilize the fixed tcp program in light of the fact that in the above situation we can see that the messages are hardcoded so we can decide the size Fixed message length

CODE:

```
class Program
  {
     private static int SendData(Socket s, byte[] data)
     {
        int total = 0;
        int size = data.Length;
        int dataleft = size;
        int sent;
        while (total < size)
        {
           sent = s.Send(data, total, dataleft, SocketFlags.None);
           total += sent;
           dataleft -= sent;
        }
        return total;
     }
     private static byte[] ReceiveData(Socket s, int size)
     {
        int total = 0;
        int dataleft = size;
        byte[] data = new byte[size];
        int recv;
        while (total < size)
        {
           recv = s.Receive(data, total, dataleft, 0);
           if (recv == 0)
           {
              data = Encoding.ASCII.GetBytes("exit ");
              break;
```

```
        }
        total += recv;
        dataleft -= recv;
    }
    return data;
}



static void Main(string[] args)
{
    byte[] data = new byte[1024];
    int sent;
    IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
    Socket server = new Socket(AddressFamily.InterNetwork,
    SocketType.Stream, ProtocolType.Tcp);
    try
    {
        server.Connect(ipep);
    }
    catch (SocketException e)
    {
        Console.WriteLine("Unable to connect to server.");
        Console.WriteLine(e.ToString());
        return;
    }
    int recv = server.Receive(data);
    string stringData = Encoding.ASCII.GetString(data, 0, recv);
    Console.WriteLine(stringData);
    sent = SendData(server, Encoding.ASCII.GetBytes("message 1"));
    sent = SendData(server, Encoding.ASCII.GetBytes("message 2"));
    sent = SendData(server, Encoding.ASCII.GetBytes("message 3"));
    sent = SendData(server, Encoding.ASCII.GetBytes("message 4"));
```

```
sent = SendData(server, Encoding.ASCII.GetBytes("message 5"));
Console.WriteLine("Disconnecting from server...");
server.Shutdown(SocketShutdown.Both);
server.Close();
```

QUESTION NO. 4 **[3+4+3 Marks]**

a. You are required to create a multimedia application for NP online Exam query session that include the following features.
   i. Instructor can upload the paper (File sharing .doc or . pdf)
   ii. Instructor can Explain the paper and Students can ask queries via chat and voice call.

b. PAF-KIET university north campus wants to transmit message to all the student about on campus examination and other updates using network functions via smtp, pop3 and IMAP. Create an application for sending the messages using mentioned protocols

A. Via chat

Server and Client

```
namespace RemoteApp
{
    public partial class ChattingMain : MetroFramework.Forms.MetroForm
    {
        string username = Chatting.name;
        public ChattingMain()
        {
            InitializeComponent();
        }

        private void ChattingMain_Load(object sender, EventArgs e)
        {
            txtuser.Text = username;
        }
```

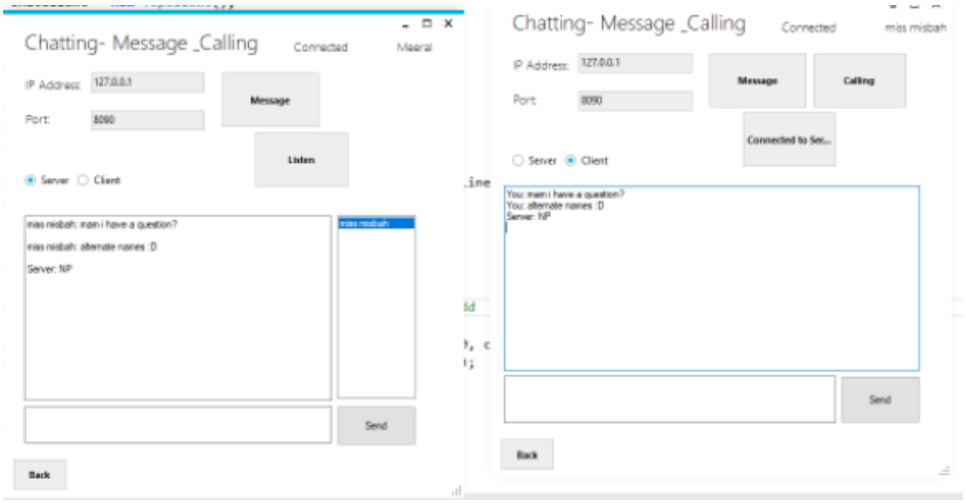
```
private void metroLabel3_Click(object sender, EventArgs e)
{
}

private void serverButton1_CheckedChanged(object sender, EventArgs e)
{
    ConnButton1.Text = "Listen";
}

private void clientButton2_CheckedChanged(object sender, EventArgs e)
{
    ConnButton1.Text = "Connected to Server";
}

private void ConnButton1_Click(object sender, EventArgs e)
{
    if (serverButton1.Checked)
    {
        ServerConnect();
        msgButton2.Visible = true;
        callButton3.Hide();
    }
    else if (clientButton2.Checked)
    {
        callButton3.Visible = true;
        msgButton2.Visible = true;
        ClientConnect();
    }
}

private TcpListener callListener;
private TcpClient callClient;
private NetworkStream stream;
Thread thread = null;
public void ServerConnect()
{
```

```
            callListener = new TcpListener(IPAddress.Parse(IpaddTextBox1.Text),
int.Parse(portTextBox2.Text));
            callListener.Start(10);
            callClient = callListener.AcceptTcpClient();
            stream = callClient.GetStream();
            metroLabel5.Text = "Connected";
            thread = new Thread(() => { Receiver(); });
            thread.Start();
            Recorder();

        }
        public void ClientConnect()
        {
            callClient = new TcpClient();

callClient.Connect(IPAddress.Parse(IpaddTextBox1.Text),Convert.ToInt32(portTextBox2.Text));
            metroLabel5.Text = "Connected";
            stream = callClient.GetStream();
        }

        public void ClientCall()
        {
            //CheckForIllegalCrossThreadCalls = false;
            thread = new Thread(() => { Receiver(); });
            thread.Start();
            Recorder();
        }


  Dictionary<string, TcpClient> lstClients = new Dictionary<string, TcpClient>();
        byte[] serverbyte = new byte[1024];

        private void msgButton2_Click(object sender, EventArgs e)
        {
            if (serverButton1.Checked)
            {
                metroPanel1.Show();
                metroPanel2.Hide();
```

```
                CheckForIllegalCrossThreadCalls = false;
                callListener.BeginAcceptTcpClient(new AsyncCallback(ClientConn), callListener);

            }
            if (clientButton2.Checked)
            {
                metroPanel2.Show();
                metroPanel1.Hide();
                CheckForIllegalCrossThreadCalls = false;
                chatClient.Connect(IPAddress.Parse(IpaddTextBox1.Text),
int.Parse(portTextBox2.Text));
                NetworkStream ns = chatClient.GetStream();
                StreamWriter sw = new StreamWriter(ns);
                sw.WriteLine("@name@" + username);
                sw.Flush();
                ns.BeginRead(clientbyte, 0, clientbyte.Length, ClientReadMsg, ns);
            }
        }

        // server send
        private void button1_Click(object sender, EventArgs e)
        {

            TcpClient serverclient;
            NetworkStream ns;
            StreamWriter sw;
            try
            {
                serverclient = (TcpClient)lstClients[listofclients.SelectedItem.ToString()];
                ns = serverclient.GetStream();
                sw = new StreamWriter(ns);
                string textToSend = "Server: " + servermsg.Text;
                sw.WriteLine(textToSend);
                serverdisplay.Text += textToSend + Environment.NewLine;
                sw.Flush();
                servermsg.Text = null;
            }
            catch (Exception)
```

```
            {
                MessageBox.Show("Select item from client list");
            }
        }

        private void ClientConn(IAsyncResult ar)
        {
            TcpListener listener = (TcpListener)ar.AsyncState;
            TcpClient client = listener.EndAcceptTcpClient(ar);
            NetworkStream ns = client.GetStream();
            object[] objectarray = new object[2];
            objectarray[0] = ns;
            objectarray[1] = client;
            ns.BeginRead(serverbyte, 0, serverbyte.Length, new AsyncCallback(getMsg),
objectarray);
            listener.BeginAcceptTcpClient(new AsyncCallback(ClientConn), listener);
        }
        private void getMsg(IAsyncResult ar)
        {
            object[] objarr = (object[])ar.AsyncState;
            NetworkStream ns = (NetworkStream)objarr[0];
            TcpClient client = (TcpClient)objarr[1];
            try
            {
                int count = ns.EndRead(ar);
                string msg = ASCIIEncoding.ASCII.GetString(serverbyte, 0, count);
                if (msg.Contains("@name@"))
                {
                    string name = msg.Replace("@name@", "");
                    lstClients.Add(name, client);
                    listofclients.Items.Add(name);
                }
                else
                {
                    serverdisplay.Text += msg + Environment.NewLine;
                }
                ns.BeginRead(serverbyte, 0, serverbyte.Length, new AsyncCallback(getMsg),
objarr);
```

```
    }
    catch (Exception) { }
}

TcpClient chatClient = new TcpClient();
byte[] clientbyte = new byte[1024];

private void serversend_Click(object sender, EventArgs e)
{
    NetworkStream ns = chatClient.GetStream();
    StreamWriter sw = new StreamWriter(ns);
    sw.WriteLine(username + ": " + clientmsg.Text);
    sw.Flush();
    clientdisplay.Text += "You: " + clientmsg.Text + Environment.NewLine;
    clientmsg.Text = "";
}

private void ClientReadMsg(IAsyncResult ar)//Client read
{
    NetworkStream ns = (NetworkStream)ar.AsyncState;  //clentt reaaddd
    int count = ns.EndRead(ar);
    clientdisplay.Text += ASCIIEncoding.ASCII.GetString(clientbyte, 0, count);
    ns.BeginRead(clientbyte, 0, clientbyte.Length, ClientReadMsg, ns);
}

private void backButton1_Click(object sender, EventArgs e)
{
    this.Hide();
    MainForm go = new MainForm();
    go.Show();
    this.Close();
}

private void txtuser_Click(object sender, EventArgs e)
{

}
```



VIA CALLING:

Server And Client

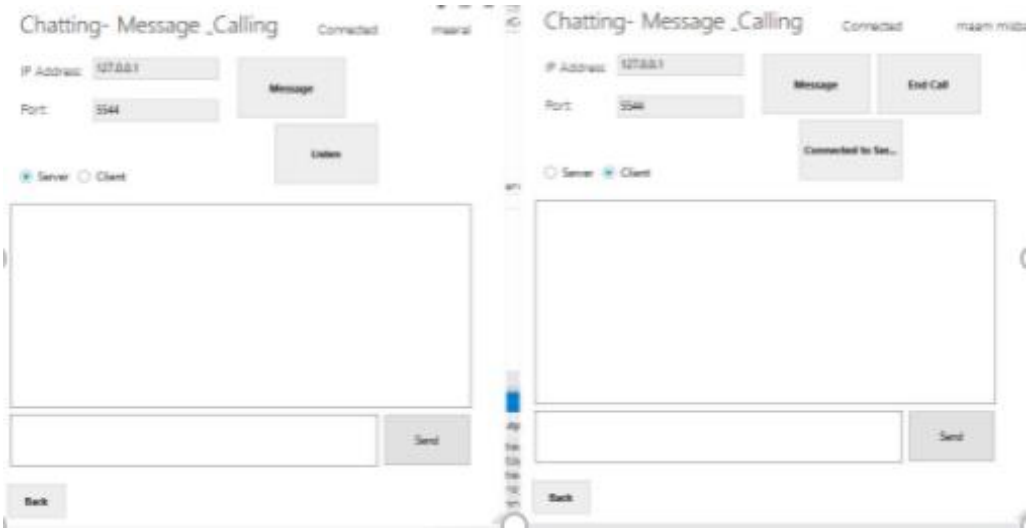
```
using System.Windows.Forms;

using NAudio.CoreAudioApi.Interfaces;

using System.Runtime.InteropServices;

using NAudio.Utils;

using NAudio.Wave;

using System.IO;


namespace RemoteApp

{

    public partial class ChattingMain : MetroFramework.Forms.MetroForm

    {

        string username = Chatting.name;

        public ChattingMain()

        {

            InitializeComponent();

        }
```

```
private void ChattingMain_Load(object sender, EventArgs e)
{
    txtuser.Text = username;
}



private void metroLabel3_Click(object sender, EventArgs e)
{
}

private void serverButton1_CheckedChanged(object sender, EventArgs e)
{
    ConnButton1.Text = "Listen";
}

private void clientButton2_CheckedChanged(object sender, EventArgs e)
{
    ConnButton1.Text = "Connected to Server";
}

private void ConnButton1_Click(object sender, EventArgs e)
{
    if (serverButton1.Checked)
    {
        ServerConnect();
        msgButton2.Visible = true;
        callButton3.Hide();
    }
    else if (clientButton2.Checked)
    {
        callButton3.Visible = true;
        msgButton2.Visible = true;
        ClientConnect();
    }
}

private TcpListener callListener;
```

```
        private TcpClient callClient;
        private NetworkStream stream;
        Thread thread = null;
        public void ServerConnect()
        {
            callListener = new TcpListener(IPAddress.Parse(IpaddTextBox1.Text),
int.Parse(portTextBox2.Text));
            callListener.Start(10);
            callClient = callListener.AcceptTcpClient();
            stream = callClient.GetStream();
            metroLabel5.Text = "Connected";
            thread = new Thread(() => { Receiver(); });
            thread.Start();
            Recorder();

        }
        public void ClientConnect()
        {
            callClient = new TcpClient();

callClient.Connect(IPAddress.Parse(IpaddTextBox1.Text),Convert.ToInt32(portTextBox2.Text));
            metroLabel5.Text = "Connected";
            stream = callClient.GetStream();
        }

        public void ClientCall()
        {
            //CheckForIllegalCrossThreadCalls = false;
            thread = new Thread(() => { Receiver(); });
            thread.Start();
            Recorder();
        }
        public WaveFormat waveFormat = new WaveFormat(148800, 1);
        WaveInEvent waveSource;
        private void Recorder()
        {
            waveSource = new WaveInEvent();
            waveSource.WaveFormat = waveFormat;
```

```
    //waveSource.StartRecording();
    waveSource.DataAvailable += new EventHandler<WaveInEventArgs>((a, e) =>
    {
        try
        {
            stream.Write(e.Buffer, 0, e.BytesRecorded);
        }
        catch
        {
            waveSource.StopRecording();
            record = false;
            read = true;
        }
    });
    waveSource.StartRecording();
}
bool read = true;
bool record = true;
private byte[] buffer = new byte[10872];
private void Receiver()
{
    var bufferedWaveProvider = new BufferedWaveProvider(waveFormat);
    var waveOut = new WaveOut();
    waveOut.Init(bufferedWaveProvider);
    waveOut.Play();
    while (true)
    {
        try
        {
            if (record)
            {
                int length = stream.Read(buffer, 0, buffer.Length);
                bufferedWaveProvider.AddSamples(buffer, 0, length);
                read = false;
            }
        }
        catch { }
    }
```

```
        }

private void callButton3_Click(object sender, EventArgs e)
        {
            if (callButton3.Text == "Calling")
            {
                ClientCall();
                callButton3.Text = "End Call";
            }
            else if (callButton3.Text == "End Call")
            {
                callButton3.Text = "Calling";
                callClient.Close();
            }
        }
```



FTP CODE:

Server

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

```
namespace ConsoleApplication5
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] data = new byte[1024];
            int sent;
            IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);

            Socket server = new Socket(AddressFamily.InterNetwork,
                            SocketType.Stream, ProtocolType.Tcp);

            try
            {
                server.Connect(ipep);
            }
            catch (SocketException e)
            {
                Console.WriteLine("Unable to connect to server.");
                Console.WriteLine(e.ToString());
                Console.ReadLine();
            }


            Bitmap bmp = new Bitmap("c:\\eek256.jpg");

            MemoryStream ms = new MemoryStream();
            // Save to memory using the Jpeg format
            bmp.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

            // read to end
            byte[] bmpBytes = ms.GetBuffer();
            bmp.Dispose();
            ms.Close();

            sent = SendVarData(server, bmpBytes);
```

```
            Console.WriteLine("Disconnecting from server...");
            server.Shutdown(SocketShutdown.Both);
            server.Close();
            Console.ReadLine();
        }

        private static int SendVarData(Socket s, byte[] data)
        {
            int total = 0;
            int size = data.Length;
            int dataleft = size;
            int sent;

            byte[] datasize = new byte[4];
            datasize = BitConverter.GetBytes(size);
            sent = s.Send(datasize);

            while (total < size)
            {
                sent = s.Send(data, total, dataleft, SocketFlags.None);
                total += sent;
                dataleft -= sent;
            }
            return total;
        }
    }
    }
}
```

Client

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

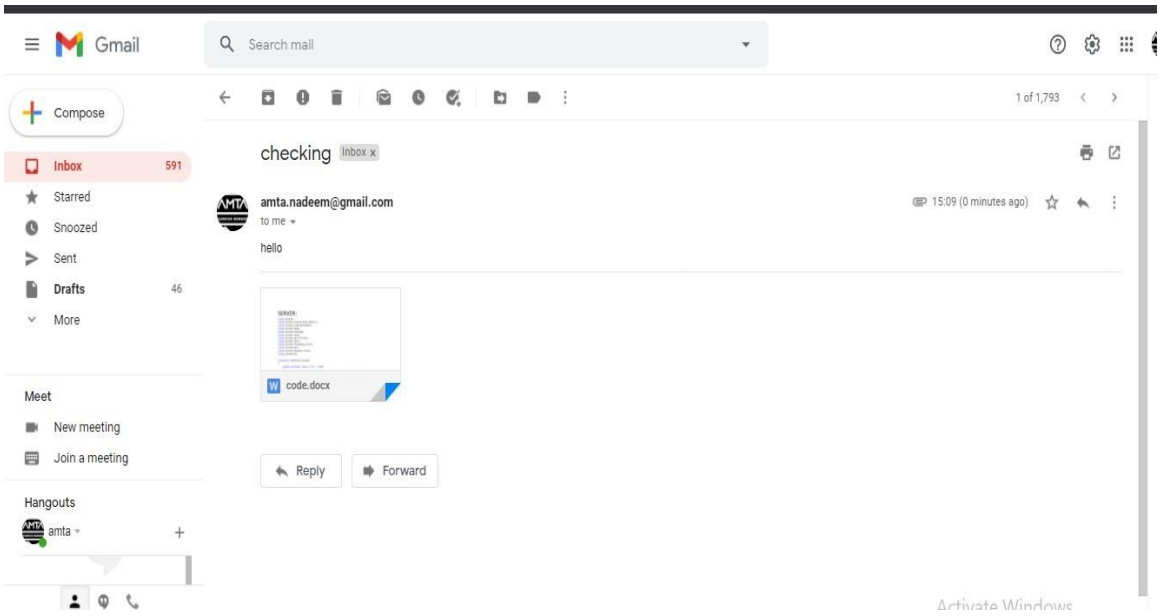
```
using System.Threading.Tasks;

namespace ConsoleApplication6
{
    class Program
    {
        static void Main(string[] args)
        {
            TcpClient client = new TcpClient();
            try
            {
                client.Connect(address, port);

                // Retrieve the network stream.
                NetworkStream stream = client.GetStream();
                MessageData data = new MessageData(imageToSend);

                IFormatter formatter = new BinaryFormatter();

                while (true)
                {
                    formatter.Serialize(stream, data);
                    Thread.Sleep(1000);
                    data.GetNewImage();
                }
            }
        }
    }
}
```

PART B

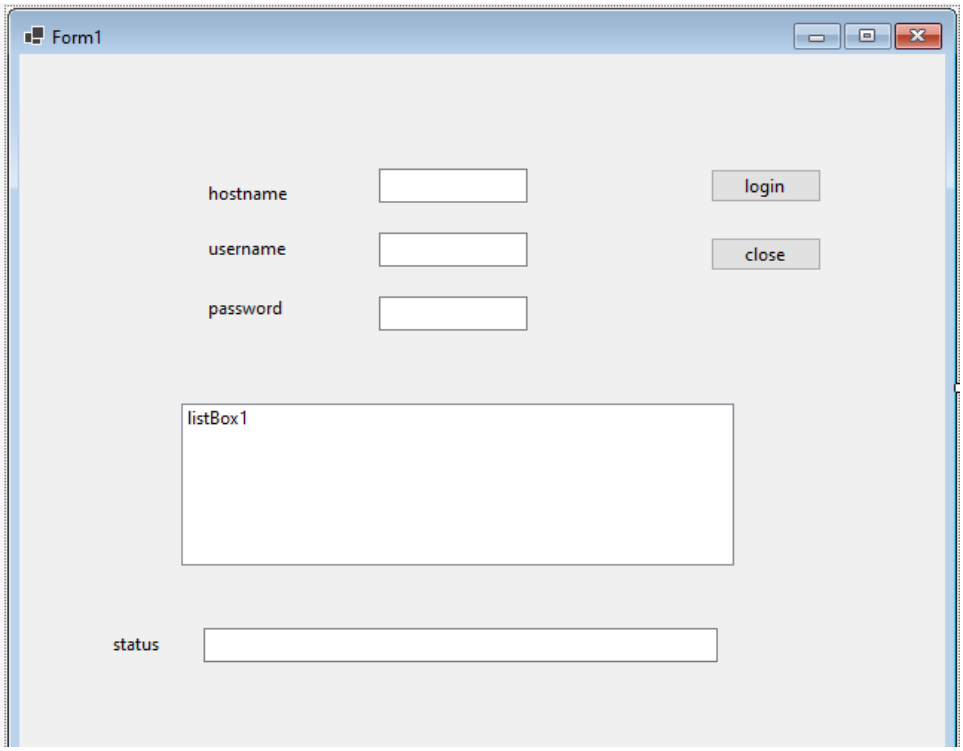
SENDING EMAIL BY SMTP

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

  private void button1_Click(object sender, EventArgs e)
  {
    try
    {


      MailMessage mails = new MailMessage();
      SmtpClient smtpserver = new SmtpClient("smtp.gmail.com");

      mails.From = new
      MailAddress("amta.nadeem@gmail.com");
      mails.To.Add(textBox1.Text);
      mails.Subject = textBox2.Text;
      mails.Body = richTextBox1.Text;
      System.Net.Mail.Attachment attachment;
      attachment = new
System.Net.Mail.Attachment("C:\\Users\\Nadeem
Butt\\Desktop\\code.docx");
      mails.Attachments.Add(attachment);
      smtpserver.Port = 587; //smtp ports: 25, 465,587,2525


      smtpserver.Credentials = new
System.Net.NetworkCredential("amta.nadeem@gmail.com", "**********");
```

```
            smtpserver.EnableSsl = true;
            smtpserver.Send(mails);
            MessageBox.Show("mail Send succesfully");
        }

        catch (Exception ex)
        {

            MessageBox.Show(ex.ToString());

        }

    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }
  }
}
```



USING POP3:




```
using System.IO;

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

      private void Form1_Load(object sender, EventArgs e)
      {
```

```
private TextBox hostname;
private TextBox username;
private TextBox password;
private TextBox status;
private ListBox messages;
private TcpClient mailclient;
private NetworkStream ns;
private StreamReader sr;
private StreamWriter sw;
public PopCheck()
{
   Text = "popcheck - A POP3 e-mail checker";
   Size = new Size(400, 380);
   Label label1 = new Label();
   label1.Parent = this;
   label1.Text = "Hostname:";
   label1.AutoSize = true;
   label1.Location = new Point(10, 33);
   hostname = new TextBox();
   hostname.Parent = this;
   hostname.Size = new Size(200, 2 * Font.Height);
   hostname.Location = new Point(75, 30);
   Label label2 = new Label();
   label2.Parent = this;
   label2.Text = "User name:";
   label2.AutoSize = true;
   label2.Location = new Point(10, 53);
   username = new TextBox();
   username.Parent = this;
   username.Size = new Size(200, 2 * Font.Height);
   username.Location = new Point(75, 50);
   Label label3 = new Label();
```

```
        label3.Parent = this;
        label3.Text = "Password:";
        label3.AutoSize = true;
        label3.Location = new Point(10, 73);
          }
password = new TextBox();
password.Parent = this;
password.PasswordChar = '*';
password.Size = new Size(200, 2 * Font.Height);
password.Location = new Point(75, 70);
Label label4 = new Label();
label4.Parent = this;
label4.Text = "Status:";
label4.AutoSize = true;
label4.Location = new Point(10, 325);
status = new TextBox();
status.Parent = this;
status.Text = "Not connected";
status.Size = new Size(200, 2 * Font.Height);
status.Location = new Point(50, 322);
messages = new ListBox();
messages.Parent = this;
messages.Location = new Point(10, 108);
messages.Size = new Size(360, 16 * Font.Height);
messages.DoubleClick += new EventHandler(getmessagesDoubleClick);
Button login = new Button();
login.Parent = this;   login.Text = "Login";
login.Location = new Point(295, 32);
login.Size = new Size(5 * Font.Height, 2 * Font.Height);
login.Click += new EventHandler(ButtonloginOnClick);
Button close = new Button();   close.Parent = this;
close.Text = "Close";   close.Location = new Point(295, 62);
```

```
close.Size = new Size(5 * Font.Height, 2 * Font.Height);
close.Click += new EventHandler(ButtoncloseOnClick);  }
void ButtonloginOnClick(object obj, EventArgs ea)  {
status.Text = "Checking for messages...";
Thread startlogin = new Thread(new ThreadStart(loginandretr));
  startlogin.IsBackground = true;   startlogin.Start();  }
void ButtoncloseOnClick(object obj, EventArgs ea)  {
  if (ns != null)   {     sw.Close();
  sr.Close();     ns.Close();
mailclient.Close();   }
Close();  }  void loginandretr()  {
string response;   string from = null;
string subject = null;   int totmessages;


try   {     mailclient = new TcpClient(hostname.Text, 110);
   } catch (SocketException)   {
status.Text = "Unable to connect to server";
     return;
}   ns = mailclient.GetStream();
sr = new StreamReader(ns);
sw = new StreamWriter(ns);
response = sr.ReadLine(); //Get opening POP3 banner
sw.WriteLine("User " + username.Text);
//Send username
sw.Flush();
response = sr.ReadLine();
if (response.Substring(0,3) == "-ER")
{
status.Text = "Unable to log into server";
return;   }
sw.WriteLine("Pass " + password.Text); //Send password
sw.Flush();   try   {
```

```
   response = sr.ReadLine();   }
catch (IOException)   {
status.Text = "Unable to log into server";
return;   }   if (response.Substring(0,4) == "-ERR")   {     s
tatus.Text = "Unable to log into server";     return;   }
sw.WriteLine("stat"); //Send stat command to get number of messages
sw.Flush();   response = sr.ReadLine();
string[] nummess = response.Split(' ');
totmessages = Convert.ToInt16(nummess[1]);
if (totmessages > 0)   {
status.Text = "you have " + totmessages + " messages";   }
else   {     status.Text = "You have no messages" ;   }
for (int i = 1; i <= totmessages; i++)   {
sw.WriteLine("top " + i + " 0"); //read header of each message
sw.Flush();     response = sr.ReadLine();
while (true)     {      response = sr.ReadLine();
if (response == ".")       break;
if (response.Length > 4)     {
if (response.Substring(0, 5) == "From:")
from = response;       if (response.Substring(0, 8) == "Subject:")
subject = response;     }    }
messages.Items.Add(i + " " + from + " " + subject);   } }
void getmessagesDoubleClick(object obj, EventArgs ea)
{   string text = (string)messages.SelectedItem;
string[] textarray = text.Split(' ');
ShowMessage sm = new ShowMessage(ns, textarray[0]);   sm.ShowDialog();
  } public static void Main()
{   Application.Run(new PopCheck());  } }
class ShowMessage :
Form {  public ShowMessage(NetworkStream ns, string messnumber)
{   StreamReader sr = new StreamReader(ns);
StreamWriter sw = new StreamWriter(ns);
```

```
string response;   Text = "Message " + messnumber;   Size = new Size(400, 380);
  ShowInTaskbar = false;   TextBox display = new TextBox();
display.Parent = this;   display.Multiline = true;
display.Dock = DockStyle.Fill;
display.ScrollBars = ScrollBars.Both;
sw.WriteLine("retr " + messnumber); //Retrieve entire message   sw.Flush();
 response = sr.ReadLine();   while (true)   {
  response = sr.ReadLine();     if (response == ".")
  break;     display.Text += response + "\r\n";   } } }
```

BY IMAP;

```
static void Main()     {
        using (var imap = new ImapClient("<ADDRESS> (e.g. imap.gmail.com)"))
{             imap.Connect();
          imap.Authenticate("<USERNAME>", "<PASSWORD>");
            imap.SelectInbox();
         int count = imap.SelectedFolder.Count;
            Console.WriteLine(" NO. |     DATE     |          SUBJECT          ");
Console.WriteLine("-------------");
            for (int number = 1; number <= count; number++)            {
                MailMessage message = imap.GetMessage(number);
              Console.WriteLine($"  {number}  |  {message.Date.ToShortDateString()}
|  {message.Subject}");            }        }    }}
```