

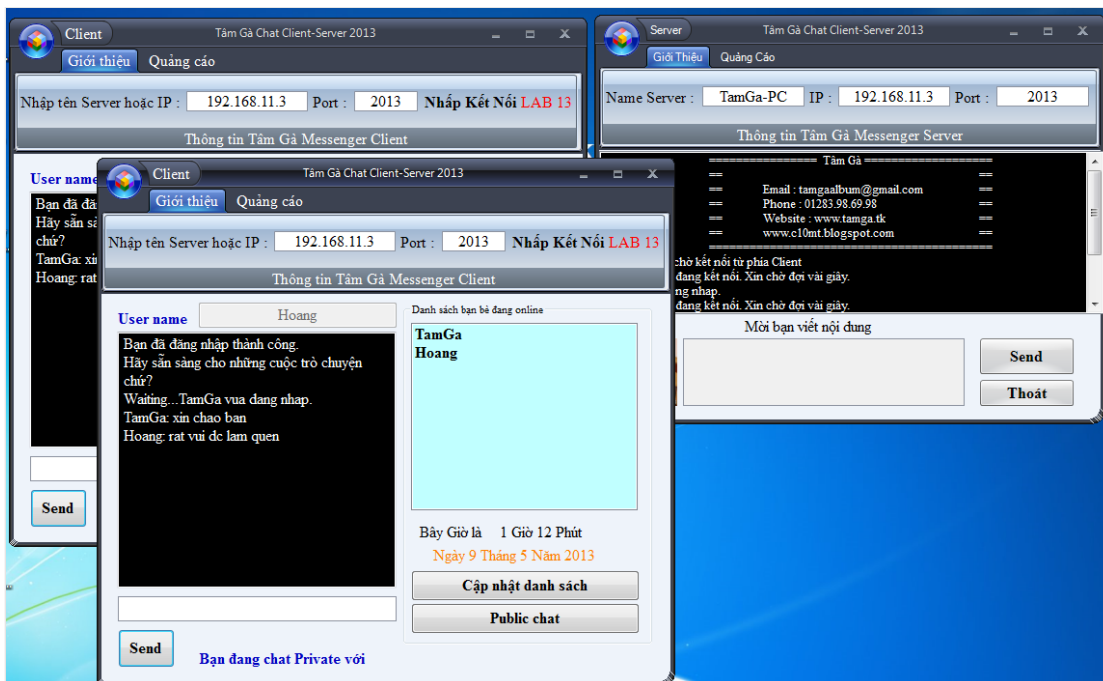
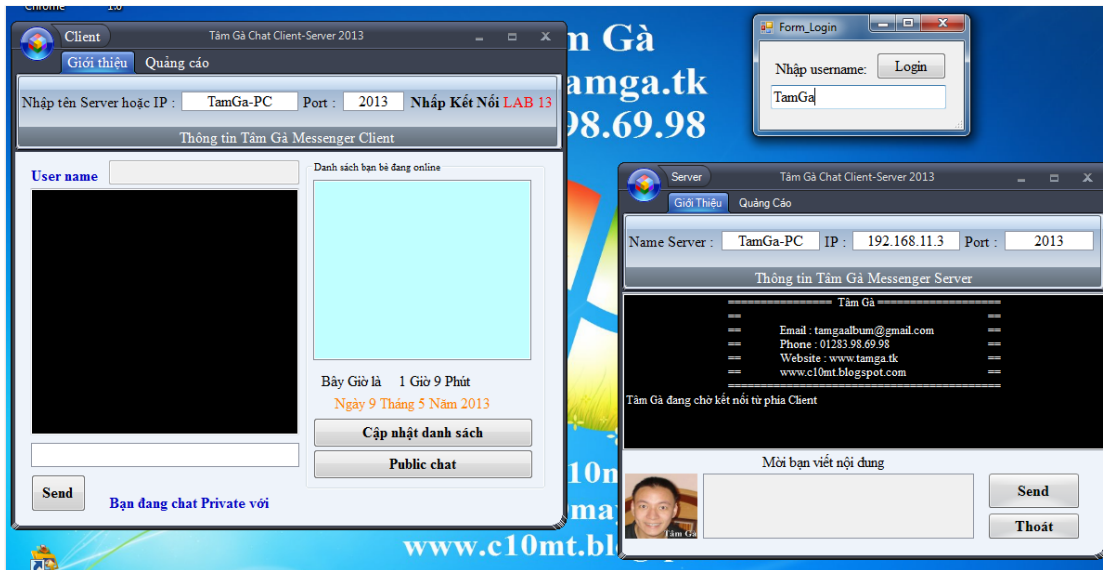
Lập Trình Mạng - LAB 13 - Windows Form - Code Chat Group, Mạng LAN

*** Bài 13 : Windows Form

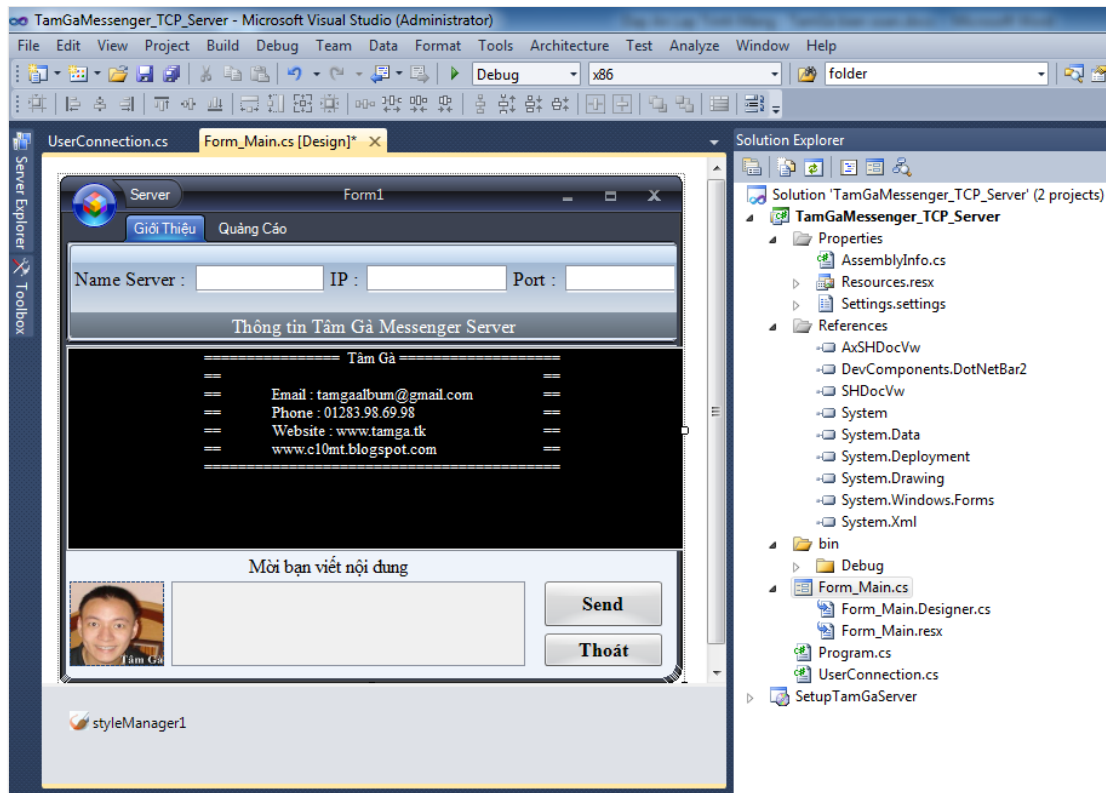
Viết chương trình chat trong mạng LAN, với yêu cầu :

- Có thể chat nhóm được (từ 2 người trở lên)
- Có thể chat từng thành viên với nhau được
- Có mã hóa đoạn chat.
- Hiện khung kết nối địa chỉ IP/Hostname và port

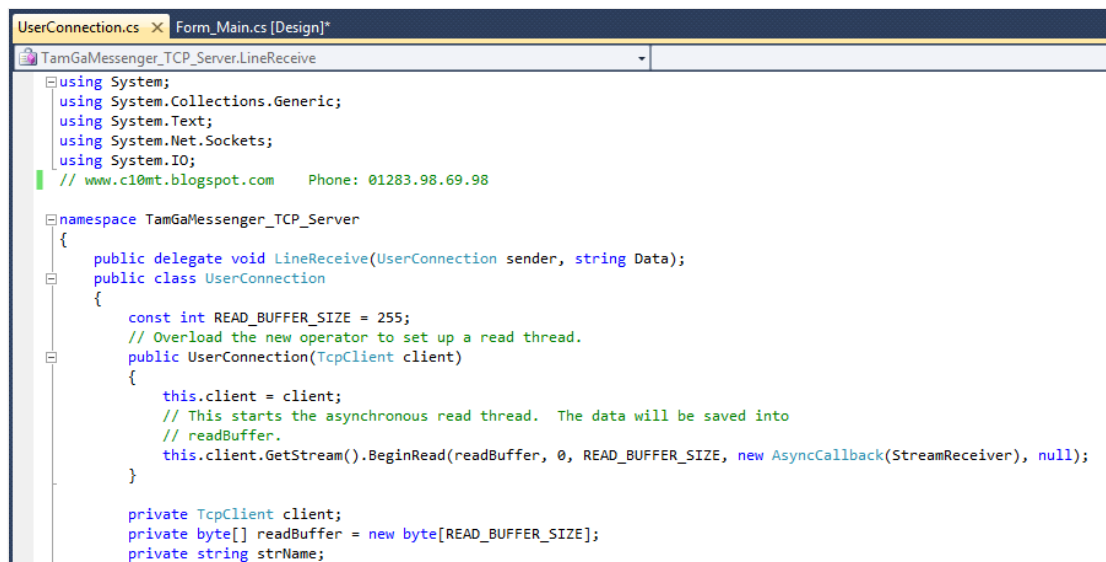
===== Đáp án mẫu như sau :



===== Và đây là code phần Server :



===== Và đây là code phần UserConnection của Server :



```

// The Name property uniquely identifies the user connection.
public string Name
{
    get
    {
        return strName;
    }
    set
    {
        strName = value;
    }
}

public event LineReceive LineReceived;

// This subroutine uses a StreamWriter to send a message to the user.
public void SendData(string Data)
{
    //lock ensure that no other threads try to use the stream at the same time.
    lock (client.GetStream())
    {
        StreamWriter writer = new StreamWriter(client.GetStream());
        writer.Write(Data + (char)13 + (char)10);
        // Make sure all data is sent now.
        writer.Flush();
    }
}
}

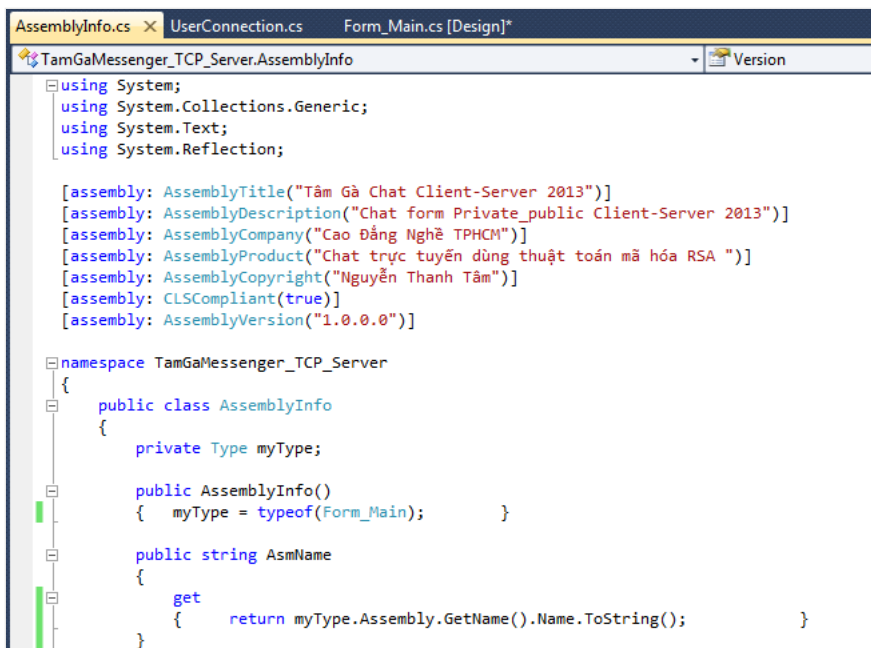
```

```

// This is the callback function for TcpClient.GetStream.Begin. It begins an
// asynchronous read from a stream.
private void StreamReceiver(IAsyncResult ar)
{
    int BytesRead;
    string strMessage;
    try
    {
        // Ensure that no other threads try to use the stream at the same time.
        lock (client.GetStream())
        {
            // Finish asynchronous read into readBuffer and get number of bytes read.
            BytesRead = client.GetStream().EndRead(ar);
        }
        // Convert the byte array the message was saved into, minus one for the
        // Chr(13).
        strMessage = Encoding.ASCII.GetString(readBuffer, 0, BytesRead - 1);
        LineReceived(this, strMessage);
        // Ensure that no other threads try to use the stream at the same time.
        lock (client.GetStream())
        {
            // Start a new asynchronous read into readBuffer.
            client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new AsyncCallback(StreamReceiver), null);
        }
    }
    catch (Exception e)
    {
    }
}
}
}

```

===== Và đây là code phần AssemblyInfo của Server :



```

AssemblyInfo.cs | UserConnection.cs | Form_Main.cs [Design]*
TamGaMessenger_TCP_Server.AssemblyInfo | Version
using System;
using System.Collections.Generic;
using System.Text;
using System.Reflection;

[assembly: AssemblyTitle("Tâm Gà Chat Client-Server 2013")]
[assembly: AssemblyDescription("Chat form Private_public Client-Server 2013")]
[assembly: AssemblyCompany("Cao Đăng Nghê TPHCM")]
[assembly: AssemblyProduct("Chat trực tuyến dùng thuật toán mã hóa RSA ")]
[assembly: AssemblyCopyright("Nguyễn Thanh Tâm")]
[assembly: CLSCompliant(true)]
[assembly: AssemblyVersion("1.0.0.0")]

namespace TamGaMessenger_TCP_Server
{
    public class AssemblyInfo
    {
        private Type myType;

        public AssemblyInfo()
        {
            myType = typeof(Form_Main);
        }

        public string AsmName
        {
            get
            {
                return myType.Assembly.GetName().Name.ToString();
            }
        }
    }
}

```



```

    public string AsmFQName
    {
        get
        {
            return myType.Assembly.GetName().FullName.ToString();
        }
    }

    public string CodeBase
    {
        get
        {
            return myType.Assembly.CodeBase;
        }
    }

    public string Copyright
    {
        get
        {
            Type at = typeof(AssemblyCopyrightAttribute);
            object[] r = myType.Assembly.GetCustomAttributes(at, false);
            AssemblyCopyrightAttribute ct = (AssemblyCopyrightAttribute)r[0];
            return ct.Copyright;
        }
    }

```

```

    public string Company
    {
        get
        {
            Type at = typeof(AssemblyCompanyAttribute);
            object[] r = myType.Assembly.GetCustomAttributes(at, false);
            AssemblyCompanyAttribute ct = (AssemblyCompanyAttribute)r[0];
            return ct.Company;
        }
    }

    public string Description
    {
        get
        {
            Type at = typeof(AssemblyDescriptionAttribute);
            object[] r = myType.Assembly.GetCustomAttributes(at, false);
            AssemblyDescriptionAttribute da = (AssemblyDescriptionAttribute)r[0];
            return da.Description;
        }
    }

```

```

    public string Product
    {
        get
        {
            Type at = typeof(AssemblyProductAttribute);
            object[] r = myType.Assembly.GetCustomAttributes(at, false);
            AssemblyProductAttribute pt = (AssemblyProductAttribute)r[0];
            return pt.Product;
        }
    }

    public string Title
    {
        get
        {
            Type at = typeof(AssemblyTitleAttribute);
            object[] r = myType.Assembly.GetCustomAttributes(at, false);
            AssemblyTitleAttribute ta = (AssemblyTitleAttribute)r[0];
            return ta.Title;
        }
    }

```

```

    public string Version
    {
        get
        {
            return myType.Assembly.GetName().Version.ToString();
        }
    }
}

```

===== Và đây là code phần main của Server :



```
Form_Main.cs* X AssemblyInfo.cs UserConnection.cs Form_Main.cs [Design]*
TamGaMessenger_TCP_Server.Form_Main PORT_NUM

/*
TamGa
www.tamga.tk www.c10mt.tk www.c10maytinh.tk
www.c10mt.blogspot.com
Phone: 01283.98.69.98 Email : tamgaalbum@yahoo.com

Bài tập : [ LAB 13 ]
Viết chương trình Chat trong Mạng LAN :
- Sử dụng giao thức TCP
- Có thể chat theo từng nhóm nhỏ.
*/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using System.Threading;
using System.Net.Sockets;
using DevComponents.DotNetBar;
using System.IO;
using System.Net;
```

```
namespace TamGaMessenger_TCP_Server
{
    public partial class Form_Main : Office2007RibbonForm
    {
        const int PORT_NUM = 2013;
        private Hashtable clients = new Hashtable();
        private TcpListener listener;
        private Thread listenerThread;
        public Form_Main()
        {
            InitializeComponent();
            AssemblyInfo ainfo = new AssemblyInfo();
            this.Text = ainfo.Title;
        }

        private void Form_Main_Load(object sender, EventArgs e)
        {
            listenerThread = new Thread(new ThreadStart(DoListen));
            listenerThread.Start();
            UpdateStatus("Tâm Gà đang chờ kết nối từ phía Client");

            TamGaKhungTenServer.Text = Dns.GetHostName();
            IPEndPoint hostname = Dns.GetHostByName(TamGaKhungTenServer.Text);
            IPAddress[] IP = hostname.AddressList;
            TamGaKhungIP.Text = IP[0].ToString();
            TamGaKhungPort.Text = PORT_NUM.ToString();
        }
    }
}
```

```
/*
Chương trình con này được sử dụng một sợi nghe nền để cho phép đọc đến
tin nhắn mà không tụt giao diện người dùng.
*/
private void DoListen()
{
    try
    {
        // Nghe cho các kết nối mới.
        listener = new TcpListener(System.Net.IPAddress.Any, PORT_NUM);
        listener.Start();
        do
        {
            // Tạo ra một kết nối người dùng mới sử dụng TcpClient trả về
            // TcpListener.AcceptTcpClient()
            UserConnection client = new UserConnection(listener.AcceptTcpClient());
            // Tạo ra một xử lý sự kiện cho phép UserConnection để giao tiếp với các cửa sổ
            client.LineReceived += new LineReceive(OnLineReceived);
            //AddHandler client.LineReceived, AddressOf OnLineReceived;
            UpdateStatus("Có một người đang kết nối. Xin chờ đợi vài giây.");
        } while (true);
    }
    catch
    {
    }
}
```



```

/* www.tamga.tk www.c10mt.tk www.c10maytinh.tk www.c10mt.blogspot.com
Chương trình con này cho biết thêm dòng vào trạng hộp danh sách
*/
private void UpdateStatus(string statusMessage)
{
    listBox_Status.Items.Add(statusMessage);
}

/*
Đây là xử lý sự kiện cho UserConnection khi nó nhận được một dòng đầy đủ.
Phân tích các cammand và các thông số và có hành động thích hợp.
*/
private void OnLineReceived(UserConnection sender, string data)
{
    string[] dataArray;
    // Phân tin nhận được chia theo "|" Phá vỡ chuỗi thành một mảng phù hợp.
    dataArray = data.Split((char)124);

```

```

// dataArray(0) is the command.
switch (dataArray[0])
{
    case "CONNECT":
        ConnectUser(dataArray[1], sender);
        break;
    case "CHAT":
        SendChat(dataArray[1], sender);
        break;
    case "DISCONNECT":
        DisconnectUser(sender);
        break;
    case "REQUESTUSERS":
        ListUsers(sender);
        break;
    default:
        UpdateStatus("Unknown message:" + data);
        break;
}
}

```

```

/* Chương trình con này sẽ kiểm tra xem nếu tên người dùng đã tồn tại trong các khách hàng Hasht
Nếu có, gửi một tin nhắn từ chối, nếu không xác nhận với một JOIN.
*/
private void ConnectUser(string userName, UserConnection sender)
{
    if (clients.Contains(userName))
    {
        ReplyToSender("REFUSE", sender);
    }
    else
    {
        sender.Name = userName;
        UpdateStatus(userName + " vừa đang nhập.");
        clients.Add(userName, sender);
        // Gửi tham gia để gửi và thông báo cho tất cả các khách hàng khác mà người gửi tham gia
        ReplyToSender("JOIN", sender);
        SendToClients("CHAT|" + "Waiting..." + sender.Name + " vừa đang nhập.", sender);
    }
}
// Chương trình con này sẽ gửi một phản ứng cho người gửi.
private void ReplyToSender(string strMessage, UserConnection sender)
{
    sender.SendData(strMessage);
}

```

```

// Chương trình con này sẽ gửi một thông điệp tới tất cả các khách hàng kèm theo, ngoại trừ người
private void SendToClients(string strMessage, UserConnection sender)
{
    UserConnection client;
    // Tất cả các mục trong các khách hàng Hashtable là UserConnection để có thể gán cho nó 1 cá
    foreach (DictionaryEntry entry in clients)
    {
        client = (UserConnection)entry.Value;
        // Loại trừ người gửi.
        if (client.Name != sender.Name)
        {
            client.SendData(strMessage);
        }
    }
}

/* www.tamga.tk www.c10mt.tk www.c10maytinh.tk www.c10mt.blogspot.com
Gửi tin nhắn trò chuyện với tất cả các khách hàng ngoại trừ người gửi.
*/
private void SendChat(string message, UserConnection sender)
{
    UpdateStatus(sender.Name + ": " + message);
    SendToClients("CHAT|" + sender.Name + ": " + message, sender);
}

```



```

/*
Chương trình con này thông báo cho các khách hàng khác mà người gửi lại trò chuyện,
và loại bỏ các tên từ các khách hàng Hashtable.
*/
private void DisconnectUser(UserConnection sender)
{
    UpdateStatus("Waiting..." + sender.Name + " đã thoát ra.");
    SendToClients("CHAT|" + "Waiting..." + sender.Name + " đã thoát ra.", sender);
    clients.Remove(sender.Name);
}

/*
Nối tất cả các tên khách hàng và gửi cho người dùng yêu cầu danh sách người dùng
*/
private void ListUsers(UserConnection sender)
{
    UserConnection client;
    string strUserList;
    UpdateStatus("Đang gửi tới " + sender.Name + " danh sách những người đang online.");
    strUserList = "LISTUSERS";
    // Tất cả các mục trong các khách hàng Hashtable là UserConnection để có thể gán cho nó 1 cá

    foreach (DictionaryEntry entry in clients)
    {
        client = (UserConnection)entry.Value;
        strUserList = strUserList + "|" + client.Name;
    }
}

```

```

// Gửi danh sách cho người gửi.
ReplyToSender(strUserList, sender);
}

// Chương trình con này sẽ gửi một thông điệp tới tất cả các khách hàng kèm theo.
private void Send(string strMessage)
{
    UserConnection client;
    // Tất cả các mục trong các khách hàng Hashtable là UserConnection để có thể gán cho nó 1 cá
    foreach (DictionaryEntry entry in clients)
    {
        client = (UserConnection)entry.Value;
        client.SendData(strMessage);
    }
}

private void buttonX1_TamGaSend_Click(object sender, EventArgs e)
{
    if (textBoxX1_Send.Text != "")
    {
        UpdateStatus("Server: " + textBoxX1_Send.Text);
        Send("BROAD|" + textBoxX1_Send.Text);
        textBoxX1_Send.Text = string.Empty;
    }
}

```

```

private void buttonX1_TamGaThoat_Click(object sender, EventArgs e)
{
    Application.Exit();
    listener.Stop();
}

private void linkLabelLapTrinhMang_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("http://c10mt.blogspot.com/2012/12/lap-trinh-mang.html");
}

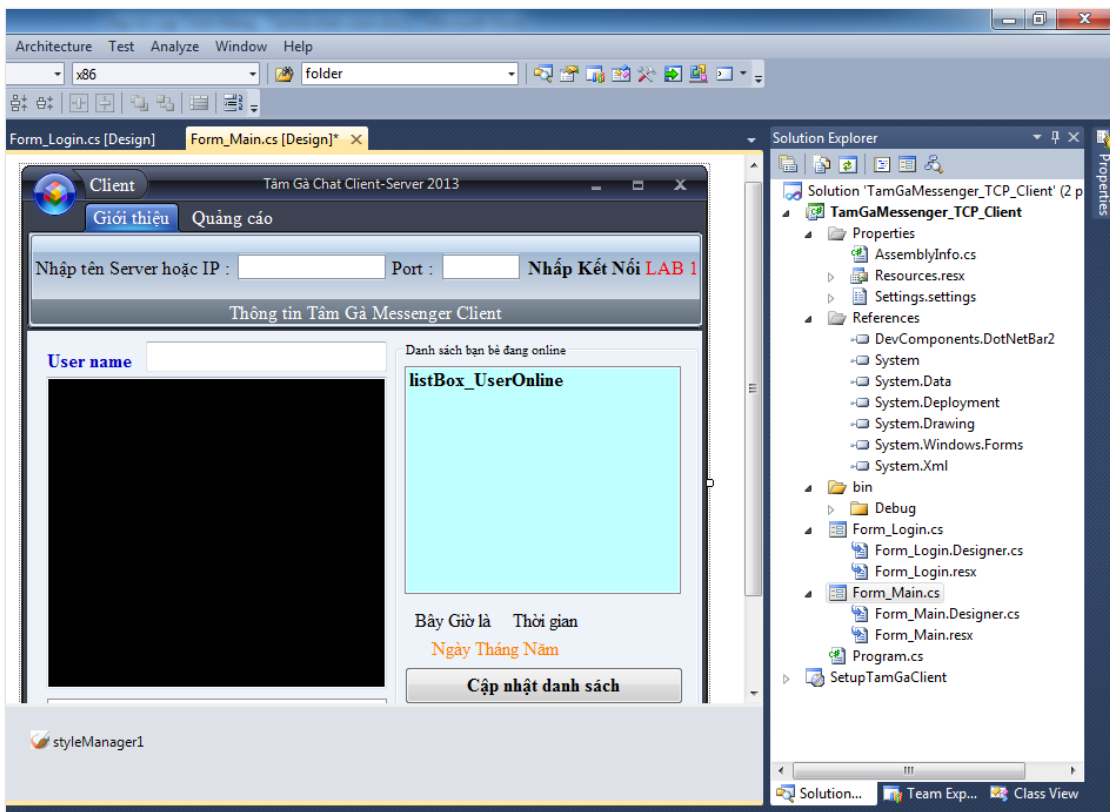
private void linkLabelC10MT_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("http://c10mt.blogspot.com");
}

// www.tamga.tk www.c10mt.tk www.c10maytinh.tk www.c10mt.blogspot.com
}

```

===== Và đây là code phần Client :





===== Và đây là code phần AssemblyInfo của Client :

```

AssemblyInfo.cs
TamGaMessenger_TCP_Client.AssemblyInfo
using System;
using System.Collections.Generic;
using System.Text;
using System.Reflection;

[assembly: AssemblyTitle("Tâm Gà Chat Client-Server 2013")]
[assembly: AssemblyDescription("Chat form Private_public Client-Server 2013")]
[assembly: AssemblyCompany("Cao Đăng Nghê TPHCM")]
[assembly: AssemblyProduct("Chat trực tuyến dùng thuật toán mã hóa RSA ")]
[assembly: AssemblyCopyright("Nguyễn Thanh Tâm")]
[assembly: CLSCompliant(true)]
[assembly: AssemblyVersion("1.0.0.0")]

namespace TamGaMessenger_TCP_Client
{
    public class AssemblyInfo
    {
        private Type myType;

        public AssemblyInfo()
        {
            myType = typeof(Form_Main);
        }

        public string AsmName
        {
            get
            {
                return myType.Assembly.GetName().Name.ToString();
            }
        }
    }
}

```




```

public string AsmFQName
{
    get
    {
        return myType.Assembly.GetName().FullName.ToString();
    }
}

public string CodeBase
{
    get
    {
        return myType.Assembly.CodeBase;
    }
}

public string Copyright
{
    get
    {
        Type at = typeof(AssemblyCopyrightAttribute);
        object[] r = myType.Assembly.GetCustomAttributes(at, false);
        AssemblyCopyrightAttribute ct = (AssemblyCopyrightAttribute)r[0];
        return ct.Copyright;
    }
}

```

```

public string Company
{
    get
    {
        Type at = typeof(AssemblyCompanyAttribute);
        object[] r = myType.Assembly.GetCustomAttributes(at, false);
        AssemblyCompanyAttribute ct = (AssemblyCompanyAttribute)r[0];
        return ct.Company;
    }
}

public string Description
{
    get
    {
        Type at = typeof(AssemblyDescriptionAttribute);
        object[] r = myType.Assembly.GetCustomAttributes(at, false);
        AssemblyDescriptionAttribute da = (AssemblyDescriptionAttribute)r[0];
        return da.Description;
    }
}

```

```

public string Product
{
    get
    {
        Type at = typeof(AssemblyProductAttribute);
        object[] r = myType.Assembly.GetCustomAttributes(at, false);
        AssemblyProductAttribute pt = (AssemblyProductAttribute)r[0];
        return pt.Product;
    }
}

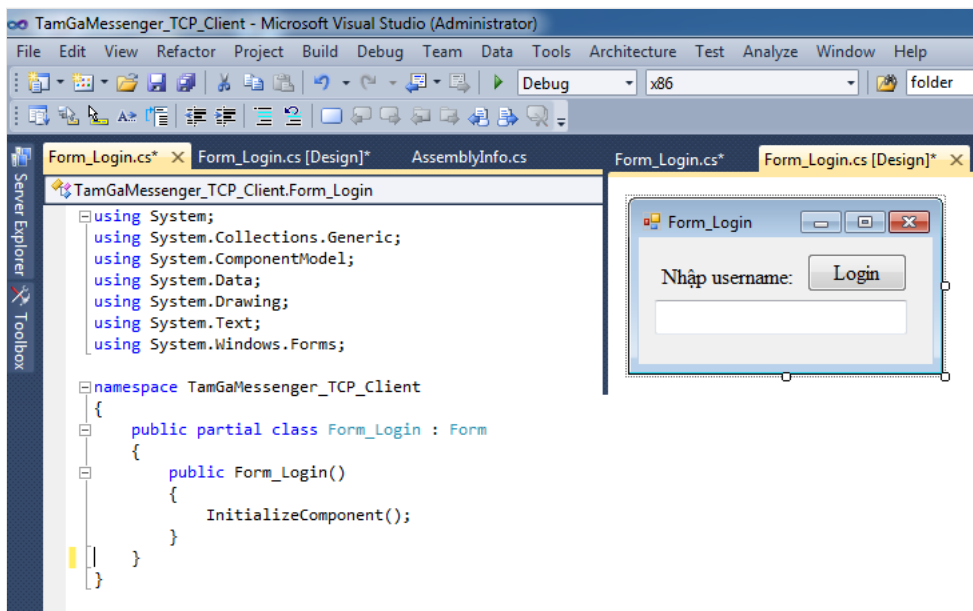
public string Title
{
    get
    {
        Type at = typeof(AssemblyTitleAttribute);
        object[] r = myType.Assembly.GetCustomAttributes(at, false);
        AssemblyTitleAttribute ta = (AssemblyTitleAttribute)r[0];
        return ta.Title;
    }
}

public string Version
{
    get
    {
        return myType.Assembly.GetName().Version.ToString();
    }
}
}

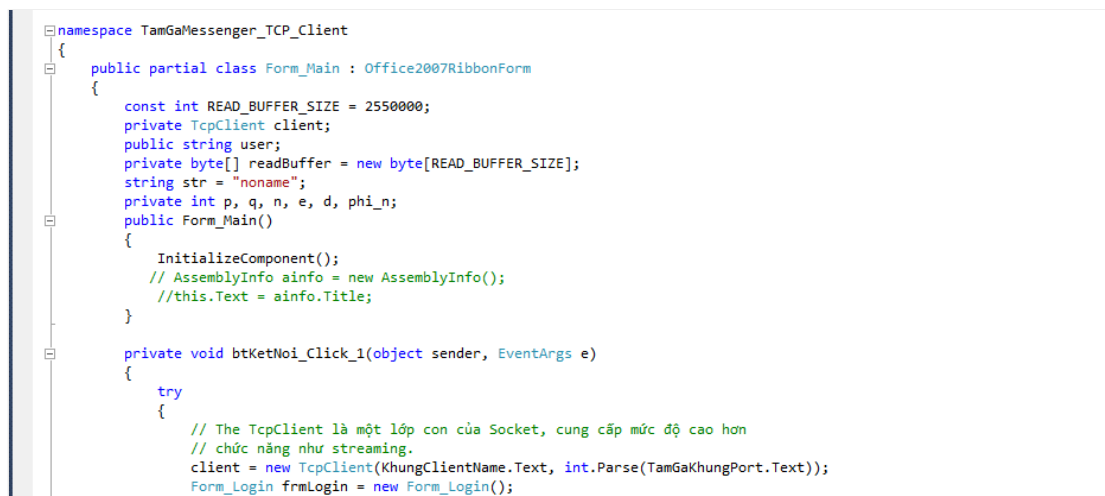
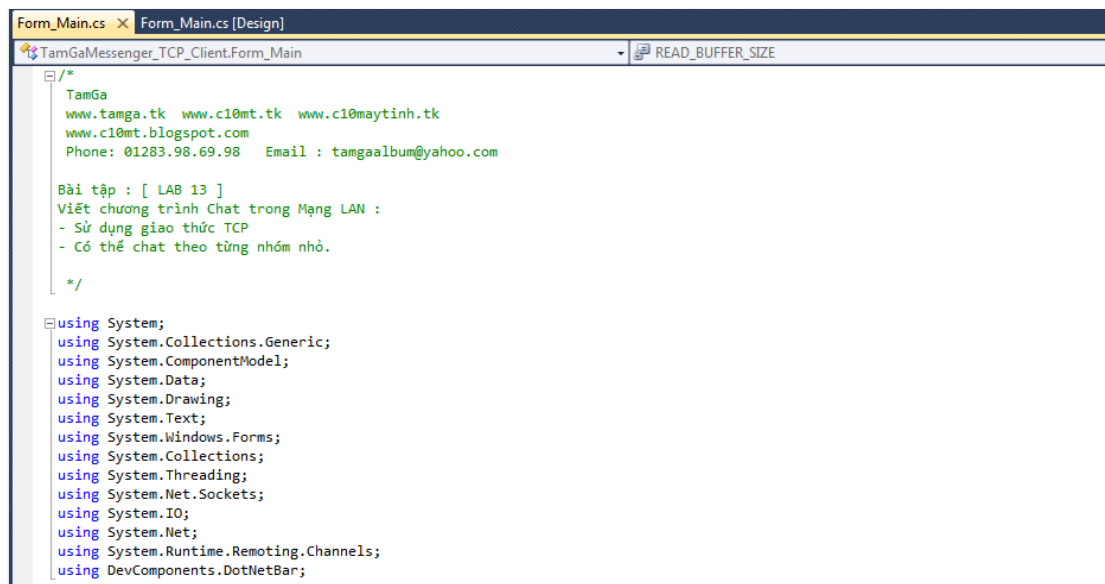
```

===== Và đây là code phần Login của Client :





===== Và đây là code phần main của Client :



```

// Bắt đầu một không đồng bộ đọc gọi DoRead để tránh tụt hậu người sử dụng
// interface.
client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new AsyncCallback(DoRead), null);
// Hãy chắc chắn rằng cửa sổ đang hiển thị trước khi xuất hiện hộp thoại kết nối.
this.Show();
AttemptLogin();
}
}
catch (Exception Ex)
{
    MessageBox.Show("Không thể kết nối với máy chủ. \r\nVui lòng thực hiện lại việc đăng nhập.",
        this.Text, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    this.Dispose();
}
}

// Đây là chức năng gọi lại cho TcpClient.GetStream.Begin để có được một không đồng bộ đọc.
private void DoRead(IAsyncResult ar)
{
    int BytesRead;
    string strMessage;

```

```

try
{
    // Kết thúc không đồng bộ đọc vào readBuffer và số trở lại của byte đọc.
    BytesRead = client.GetStream().EndRead(ar);
    if (BytesRead < 1)
    {
        // Nếu không byte đã được đọc máy chủ có gần. Vô hiệu hóa cửa sổ nhập.
        MarkAsDisconnected();
        return;
    }
    // Chuyển đổi các mảng byte thông điệp đã được lưu vào, trừ hai cho
    // Chr(13) and Chr(10)
    strMessage = Encoding.ASCII.GetString(readBuffer, 0, BytesRead - 2);
    ProcessCommands(strMessage);
    // Bắt đầu không đồng bộ mới đọc vào readBuffer.
    client.GetStream().BeginRead(readBuffer, 0, READ_BUFFER_SIZE, new AsyncCallback(DoRead), null);
}
catch (Exception e)
{
    MarkAsDisconnected();
}
}

```

```

// Khi máy chủ ngắt kết nối, ngăn chặn tin nhắn trò chuyện tiếp tục được gửi đi.
private void MarkAsDisconnected()
{
    TextBoxX1_TamGaSend.ReadOnly = true;
    button_Send.Enabled = false;
}

// Xử lý các lệnh nhận được từ máy chủ, và có hành động thích hợp.
private void ProcessCommands(string strMessage)
{
    string[] dataArray;
    // Phân tin nhắn được chia theo "|" Phá vỡ chuỗi thành một mảng phù hợp.
    dataArray = strMessage.Split((char)124);
    // dataArray[0] is the command.
    switch (dataArray[0])
    {
        case "JOIN":
            // Máy chủ nhận đăng nhập.
            DisplayText("Bạn đã đăng nhập thành công. "
                + "\r\nHãy sẵn sàng cho những cuộc trò chuyện chứ?" + (char)13 + (char)10);
            break;
        case "CHAT":
            // Nhận được tin nhắn trò chuyện, hiển thị nó.
            if (dataArray[1].Substring(0, 10) == "Waiting...")
                DisplayText(dataArray[1] + (char)13 + (char)10);

```

```

            else
            {
                string data = dataArray[1];
                string subStringYes = data.Substring(data.Length - user.Length, user.Length);
                string subStringNo = data.Substring(data.Length - "noname".Length, "noname".Length);
                if (subStringYes == user)
                    DisplayText(getString(data.Substring(0, data.Length - user.Length)) + (char)13 + (char)10);
                else if (subStringNo == "noname")
                    DisplayText(getString(data.Substring(0, data.Length - "noname".Length)) + (char)13 + (char)10);
            }
            break;
        case "REFUSE":
            // Máy chủ từ chối đăng nhập với tên người dùng, hãy thử đăng nhập với một tên khác.
            AttemptLogin();
            break;
        case "LISTUSERS":
            // Máy chủ gửi một danh sách người dùng.
            ListUsers(dataArray);
            break;
        case "BROAD":
            // Máy chủ gửi một thông điệp phát sóng.
            DisplayText("Máy chủ: " + dataArray[1] + (char)13 + (char)10);
            break;
    }
}
}

```



```
// Viết văn bản vào hộp văn bản đầu ra.
private void DisplayText(string text)
{
    textBox_Status.AppendText(text);
}

// Bật lên một hộp thoại người dùng kết nối và gửi một yêu cầu người sử dụng tin nhắn để đăng nhập vào trò chuyện.
void AttemptLogin()
{
    Form_Login frmLogin = new Form_Login();
    frmLogin.StartPosition = FormStartPosition.CenterParent;
    frmLogin.ShowDialog(this);
    SendData("CONNECT" + frmLogin.textBox_UserFormLogin.Text);
    frmLogin.Dispose();
    user = frmLogin.textBox_UserFormLogin.Text;
    txKhungTenChat.Text = user;
}

//Dùng RSA
private string GetString(string str)
{
    int bre = 0;
    string s1 = null;
    string s2 = null;
    string s = null;
    char[] getChar = new char[str.Length];
    getChar = str.ToCharArray();

    for (int i = 0; i < str.Length; i++)
    {
        if (getChar[i] == ':')
        {
            bre = i + 1;
            break;
        }
    }
    for (int k = 0; k < bre; k++)
    {
        s1 += getChar[k].ToString();
    }
    for (int j = bre; j < str.Length; j++)
    {
        s2 += getChar[j].ToString();
    }
    s2 = Decipher(s2);
    s = s1 + " " + s2;
    return s;
}

private string Decipher(string str)
{
    //str = GetString(str);
    string rtbChuoiKiTu = str.Trim() + " ";
    int chieuDaiChuoi = rtbChuoiKiTu.Length;
    char[] rtbMangKiTu;
    rtbMangKiTu = new char[chieuDaiChuoi];
    int[] rtbMangSo;
    rtbMangSo = new int[chieuDaiChuoi];
    rtbMangKiTu = rtbChuoiKiTu.ToCharArray();
    string s = "";
    int count = 0;
    int i = 0;
    for (i = 0; i < chieuDaiChuoi; i++)
    {
        if (rtbMangKiTu[i] != ' ')
        {
            s += rtbMangKiTu[i];
        }
        else
        {
            rtbMangSo[count] = int.Parse(s);
            count++;
            s = "";
        }
    }
}
```



```

char[] rtbMang;
rtbMang = new char[chieuDaiChuoiv];
int dd = rtbMangSo[0];
int ee = rtbMangSo[1];
int nn = rtbMangSo[2];
for (i = 3; i < count; i++)
{
    rtbMangSo[i] = (rtbMangSo[i] ^ dd) % nn;
    rtbMangSo[i] = (rtbMangSo[i] ^ ee) % nn;
    rtbMangSo[i] = (rtbMangSo[i] ^ dd) % nn;
    rtbMang[i] = (char)rtbMangSo[i];
    s += rtbMang[i];
}
return s;
}

// Chương trình con này cho biết thêm một danh sách các người sử dụng hộp danh sách.
private void ListUsers(string[] users)
{
    int I;

    for (I = 1; I <= (users.Length - 1); I++)
    {
        listBox_UserOnline.Items.Add(users[I]);
    }
}

```

```

// Sử dụng một StreamWriter để gửi tin nhắn đến máy chủ.
private void SendData(string data)
{
    StreamWriter writer = new StreamWriter(client.GetStream());
    writer.Write(data + (char)13);
    writer.Flush();
}

private string Ecrypt(string str)
{
    taoKhoa();
    int len = str.Length;
    char[] mangKiTu = new char[len];
    mangKiTu = str.ToCharArray();
    int[] mangAscii = new int[len];
    for (int i = 0; i < len; i++)
        mangAscii[i] = (int)mangKiTu[i];
    for (int i = 0; i < len; i++)

        mangAscii[i] = (mangAscii[i] ^ e) % n;          //Mã hóa từng kí tự trong chuỗi

    string str1 = "";          // Gán vào một chuỗi số khác
    for (int i = 0; i < len; i++)
        str1 += (mangAscii[i] + " ");
    return d.ToString() + " " + e.ToString() + " " + n.ToString() + " " + str1;
}

```

```

//Hàm tạo các khóa
private void taoKhoa()
{
    //Tạo hai số nguyên tố ngẫu nhiên khác nhau
    do
    {
        p = soNgauNhan();
        q = soNgauNhan();
    }
    while (p == q || !kiemTraNguyenTo(p) || !kiemTraNguyenTo(q));

    //Tính n=p*q
    n = p * q;

    //Tính Phi(n)=(p-1)*(q-1)
    phi_n = (p - 1) * (q - 1);

    //Tính e là một số ngẫu nhiên có giá trị 0< e <phi(n) và là số nguyên tố cùng nhau với Phi(n)
    do
    {
        Random rd = new Random();
        e = rd.Next(2, phi_n);
    }
    while (!NguyenToCungNhan(e, phi_n));
}

```



```

//Tính d
d = 0;
int i = 2;
while (((1 + i * phi_n) % e) != 0 || d <= 0)
{
    i++;
    d = (1 + i * phi_n) / e;
}
}
//Hàm tạo số ngẫu nhiên từ 2->10000
private int soNgauNhan()
{
    Random rd = new Random();
    return rd.Next(11, 101);
}

//Hàm kiểm tra nguyên tố
private bool kiểmTraNguyenTo(int i)
{
    bool kiểmtra = true;
    for (int j = 2; j < i; j++)
        if (i % j == 0)
            kiểmtra = false;
    return kiểmtra;
}

```

```

//Hàm kiểm tra hai số nguyên tố cùng nhau
private bool nguyenToCungNhanh(int a, int b)
{
    bool kiểmtra = true;
    for (int i = 2; i < a; i++)
        if (a % i == 0 && b % i == 0)
            kiểmtra = false;
    return kiểmtra;
}

// Gửi máy chủ tin nhắn ngắt kết nối
private void Form_Main_Closing(object sender, System.ComponentModel.CancelEventArgs e) //base.Closing;
{
    // Chỉ gửi nếu máy chủ vẫn chạy.
    if (button_Send.Enabled == true)
    {
        SendData("DISCONNECT");
    }
}

private void button_Send_Click_1(object sender, EventArgs e)
{
    if (TextBoxX1_TamGaSend.Text != "")
    {
        DisplayText(user + ": " + TextBoxX1_TamGaSend.Text + (char)13 + (char)10);
        SendData("CHAT|" + Ecrypt(TextBoxX1_TamGaSend.Text) + str);
        TextBoxX1_TamGaSend.Text = string.Empty;
    }
}

```

```

private void button_CapNhatDanhSach_Click_1(object sender, EventArgs e)
{
    listBox_UserOnline.Items.Clear();
    SendData("REQUESTUSERS");
}

private void button_PublicChat_Click_1(object sender, EventArgs e)
{
    str = "noname";
    label_Private.Text = "Bạn đang chat public !!!";
}

private void linkLabelC10MT_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("http://c10mt.blogspot.com");
}

private void linkLabelLapTrinhMang_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("http://c10mt.blogspot.com/2012/12/lap-trinh-mang.html");
}

```

```

private void listBox_UserOnline_SelectedIndexChanged(object sender, EventArgs e)
{
    str = listBox_UserOnline.Text;
    label_Private.Text = "Bạn đang chat private với: " + str;
}

int day = DateTime.Now.Day;
int month = DateTime.Now.Month;
int year = DateTime.Now.Year;
int gio = DateTime.Now.Hour;
int phut = DateTime.Now.Minute;

private void Form_Main_Load(object sender, EventArgs e)
{
    ThoiGian.Text = gio + " Giờ " + phut + " Phút ";
    NgayThangNam.Text = "Ngày " + day + " Tháng " + month + " Năm " + year;
}
}

```



Lập Trình Mạng - LAB 13 - Windows Form - Code Chat Group, Mạng LAN

Bạn đang xem một trong các bài viết tại Chuyên Mục **C#Lập Trình Mạng**. Và đây là địa chỉ link bài viết <http://www.c10mt.com/2012/05/tamga-ltm-lab13.html> . Tâm Gà xin cảm ơn bạn đã theo dõi bài viết này. Đừng quên nhấn **LIKE** và **Chia Sẻ** để ủng hộ Tâm Gà nếu bài viết có ích !

Tuesday, May 1, 2012 **DMCA** **PROTECTED**

Chia sẻ:    

XEM THÊM

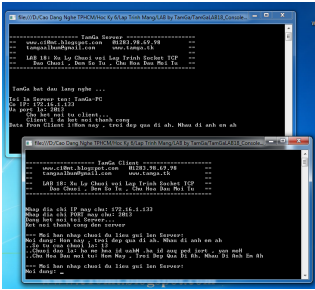
Danh sách: Điểm danh - Cần Thi - Điểm chuyên cần									
TT	Mã số SV-HS	Họ và tên SV-HS	Điểm học	Điểm chuyên cần	Điểm tổng	Điểm trung bình	Điểm xếp loại	Điểm xếp loại	Điểm xếp loại
1	C10050463	Trần Quốc Anh	18	18	36	18	0,2	0,2	0,2
2	C10050362	Nguyễn Thành K Đăng	18	18	36	18	0,2	0,2	0,2
3	C10051133	Nguyễn Xuân K Thanh	17	17	34	17	0,1	0,1	0,1
4	C10051130	Vũ Đình K Hoàng	17	17	34	17	0,1	0,1	0,1
5	C10051126	Nguyễn Quốc K Huy	17	17	34	17	0,1	0,1	0,1
6	C10050636	Chung Quốc K Đạt	17	17	34	17	0,1	0,1	0,1
7	C10050721	Nguyễn Trọng K Giang	17	17	34	17	0,1	0,1	0,1
8	C10050626	Châu D K Hòa	17	17	34	17	0,1	0,1	0,1
9	C10050730	Nguyễn Phú K Hà	17	17	34	17	0,1	0,1	0,1
10	C10050729	Vũ Phi K Hoàng	17	17	34	17	0,1	0,1	0,1
11	C10051050	Bàng Quốc K Huy	17	17	34	17	0,1	0,1	0,1
12	C10050465	Lê Bình K Hoa	17	17	34	17	0,1	0,1	0,1
13	C10050464	Trần Đình K Khoa	17	17	34	17	0,1	0,1	0,1
14	C10050652	Lê Đức K Anh	16	16	32	16	0,1	0,1	0,1
15	C10050655	Chung Quốc K Long	16	16	32	16	0,1	0,1	0,1
16	C10051151	Nguyễn Khôi Nam	16	16	32	16	0,1	0,1	0,1
17	C10051052	Vũ Quốc K Nam	16	16	32	16	0,1	0,1	0,1
18	C10050388	Đặng Thanh K Nhân	17	17	34	17	0,1	0,1	0,1
19	C10051124	Nguyễn Văn K Hoàng	16	16	32	16	0,1	0,1	0,1
20	C10050466	Nguyễn Văn K Phú	16	16	32	16	0,1	0,1	0,1
21	C10051198	Quách Văn K Phú	16	16	32	16	0,1	0,1	0,1
22	C10050383	Nguyễn Thanh K Phong	16	16	32	16	0,1	0,1	0,1
23	C10050393	Trần Văn K Phú	16	16	32	16	0,1	0,1	0,1
24	C10051097	Nguyễn K K	16	16	32	16	0,1	0,1	0,1
25	C10050628	Nguyễn Xuân K Quang	16	16	32	16	0,1	0,1	0,1
26	C10051137	Nguyễn Thanh Mạnh K Quý	16	16	32	16	0,1	0,1	0,1
27	C10051044	Nguyễn Thanh K Quý	16	16	32	16	0,1	0,1	0,1
28	C10051056	Trần Hữu K Tâm	16	16	32	16	0,1	0,1	0,1
29	C10050630	Nguyễn Trọng K Tuấn	16	16	32	16	0,1	0,1	0,1
30	C10050739	Nguyễn Hoàng K Tà	16	16	32	16	0,1	0,1	0,1
31	C10050739	Nguyễn Hòa Mạnh K Tuấn	16	16	32	16	0,1	0,1	0,1
32	C10050738	Trần Mạnh K Tuấn	16	16	32	16	0,1	0,1	0,1
33	C10050620	Tô Hoàng K Tuấn	16	16	32	16	0,1	0,1	0,1
34	C10050625	Lê Hoàng K Tuấn	16	16	32	16	0,1	0,1	0,1
35	C10050606	Nguyễn Quốc K Thanh	17	17	34	17	0,1	0,1	0,1

[Lập Trình Mạng] Danh Sách
Cần Thi và Điểm Chuyên Cần
[update 26/04/2013]
01-5-2013



Lập Trình Mạng - LAB 19 -
Windows Form & Console C# -
Máy Tính Đơn Giản

01-5-2012



Lập Trình Mạng - LAB 18 -
Console C# - Nhập Đảo Ngược
Đếm Chuỗi

01-5-2012



Lậ:
Cc:
01- i

