Time: 1-2 hours; open book
Answer at least 2 questions out of 3, complete using vb.net/c# (Visual Studio) or other programming languages

Note: Please find the entire source code of my work in my Github
https://github.com/jercylew/tech_interview

Q1: Design a product catalogue with products (name, price, description), and n-level and multiple categories and manufacturer (name, logo). Draw normalized table structure with primary & foreign keys and write SQL to retrieve all n-level category products recursively, expected output:
Books – Philosophy – Metaphysics
Books – Philosophy – Confucianism - Mencius
Books – Literature – Lin Yutang
Software – Utilities – File Management

A1:

1) Create database and tables

```
-- ==========================================
-- Create database for catelogue product
-- ==========================================
USE master
GO

-- DECLARE @dbname nvarchar(128)
-- SET @dbname = N'product_catalogue'
-- GO

-- Drop the database if it already exists
IF  EXISTS (
     SELECT name
          FROM sys.databases
          WHERE name = 'product_catalogue'
)
DROP DATABASE product_catalogue

CREATE DATABASE product_catalogue
USE product_catalogue
GO

-- Create Catelogue table
IF OBJECT_ID('Catelogue', 'U') IS NOT NULL
  DROP TABLE Catelogue

CREATE TABLE Catelogue (
    CatelogueID int NOT NULL PRIMARY KEY,
    CatelogueName varchar(128) NOT NULL,
    ParentCatelogueID int FOREIGN KEY REFERENCES Catelogue (CatelogueID)
);
GO

-- Create Product table
IF OBJECT_ID('Product', 'U') IS NOT NULL
  DROP TABLE Product
```

```sql
CREATE TABLE Product (
    ProductID int NOT NULL PRIMARY KEY,
    ProductName varchar(128) NOT NULL,
     Price float,
     ProductDescription varchar(256),
    CatelogueID int FOREIGN KEY REFERENCES Catelogue (CatelogueID)
);
GO

-- Create Manufacture table
IF OBJECT_ID('Manufacture', 'U') IS NOT NULL
  DROP TABLE Manufacture

CREATE TABLE Manufacture (
    ManufactureID int NOT NULL PRIMARY KEY,
    ManufactureName varchar(128) NOT NULL,
);
GO
```

2) Adding test data and query category product

```
use product_catalogue

-- insert test data
if not exists (
        select * from Catelogue
)
begin
    insert into Catelogue values (1000, 'Books', null)
    insert into Catelogue values (1001, 'Philosophy', 1000)
    insert into Catelogue values (1003, 'Confucianism', 1001)
    insert into Catelogue values (1005, 'Literature', 1000)
    insert into Catelogue values (1007, 'Software', null)
    insert into Catelogue values (1008, 'Utilities', 1007)

    insert into Product values (1002, 'Metaphysics', 25.50, 'A introduction to
metaphysics', 1001)
    insert into Product values (1004, 'Mencius', 35.20, 'A introduction to
mencius', 1003)
    insert into Product values (1006, 'Lin Yutang', 65.50, 'A introduction to
lin yutang', 1005)
    insert into Product values (1009, 'File Management', 125.50, 'A
introduction to file management', 1008)
end
go
select * from Product
```

Results | Messages

	ProductID	ProductName	Price	ProductDescription	CatelogueID
1	1002	Metaphysics	25.5	A introduction to metaphysics	1001
2	1004	Mencius	35.2	A introduction to mencius	1003
3	1006	Lin Yutang	65.5	A introduction to lin yutang	1005
4	1009	File Management	125.5	A introduction to file management	1008

Query executed successfully. | MS-20160717HYFS\SQLEXPRESS ... | MS-20160717HYFS\Admini...

```
select * from Catelogue
```

Results | Messages

	CatelogueID	CatelogueName	ParentCatelogueID
1	1000	Books	NULL
2	1001	Philosophy	1000
3	1003	Confucianism	1001
4	1005	Literature	1000
5	1007	Software	NULL
6	1008	Utilities	1007

```
-- query the n-level category product recursively
with n(CatelogueID, CatelogueName, ParentcatelogueID, CatelogueLevel,
Concatenador) as
(
    select CatelogueID, CatelogueName, ParentcatelogueID, 1 as CatelogueLevel,
convert(varchar(max), CatelogueName) as Concatenador
    from Catelogue
    where ParentcatelogueID is null
        union all
    select m.CatelogueID, m.CatelogueName, m.ParentcatelogueID,
CatelogueLevel+1, n.Concatenador+' - '+convert(varchar(max), m.CatelogueName)
as Concatenador
    from Catelogue as m, n
    where n.CatelogueID = m.ParentcatelogueID
)
select  n.Concatenador + ' - ' + Product.ProductName from Product, n where
Product.CatelogueID = n.CatelogueID order by n.Concatenador asc
```

Results | Messages

	(No column name)
1	Books - Literature - Lin Yutang
2	Books - Philosophy - Metaphysics
3	Books - Philosophy - Confucianism - Mencius
4	Software - Utilities - File Management

Query executed successfully. | MS-20160717HYFS\SQLEXPRESS ... | MS-20

Q2. Write an ASP.net web site (or in other programming languages) with a "contact us" web form to send email to hr@reasonables.com. Use web@reasonables.com as SENDER, get FROM from contact us form. Use local IIS SMTP as SMTP server. The following email header is expected:

```
Sender: <web@reasonables.com>
Return-Path: <web@reasonables.com>
…....

Date: Sun, 22 Mar 2009 16:16:12 +0800
From: John Smith<johnsmith@reasonables.com>
Reply-To:<web@reasonables.com>
To: <hr@reasonables.com>
Subject: Realizing Value from IT Investment
```

A2

1) Page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Contact Us.aspx.cs"
Inherits="TestWeb.Contact_Us" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        #TextArea1 {
            height: 111px;
            width: 314px;
        }

        body {font-family: Arial, Helvetica, sans-serif;}
        * {box-sizing: border-box;}

        input[type=text], select, textarea {
            padding: 12px;
            border: 1px solid #ccc;
            border-radius: 4px;
            box-sizing: border-box;
            margin-top: 6px;
            margin-bottom: 16px;
            resize: vertical;
        }

        input[type=submit] {
            background-color: #2471A3; /* #4CAF50 */
            color: white;
            padding: 12px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
```

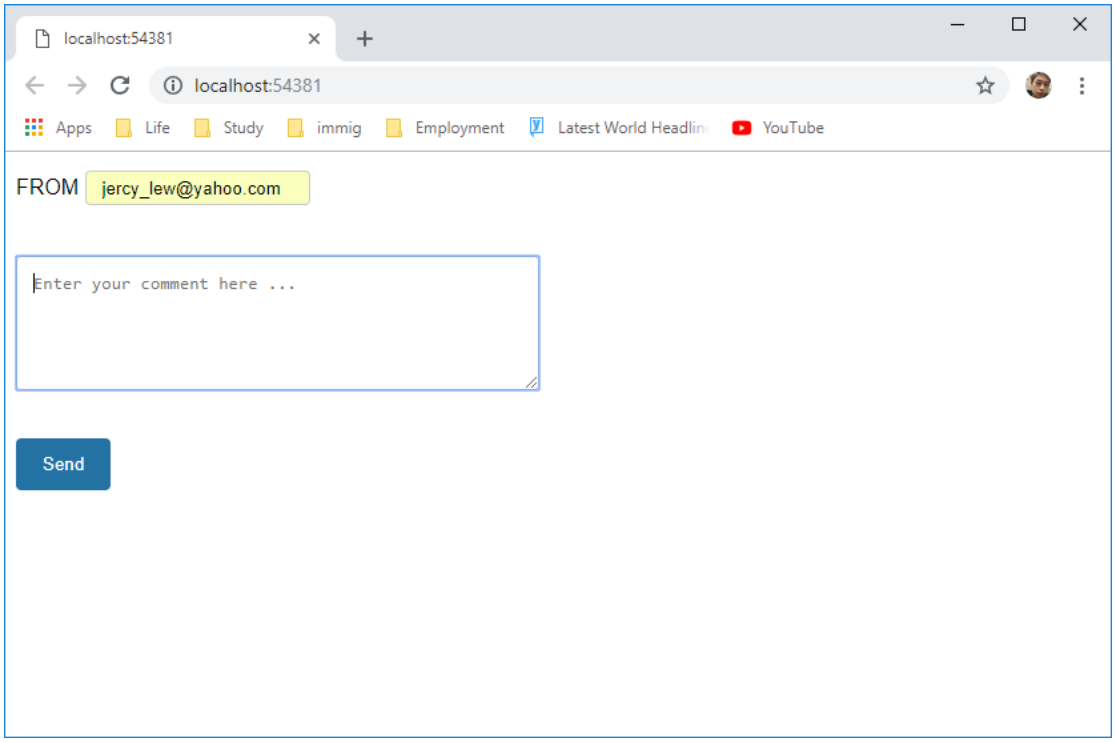
```
        input[type=submit]:hover {
            background-color: #45a049;
        }

        .container {
            border-radius: 5px;
            background-color: #f2f2f2;
            padding: 20px;
        }
    </style>
</head>
<body style="height: 172px">
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="m_lblFrom" for="m_txtFrom" runat="server"
Text="FROM"></asp:Label>
            <asp:TextBox ID="m_txtFrom" runat="server" Width="169px"
Height="22px" placeholder="example@domain.com"></asp:TextBox>
        </div>
        <p>
            <asp:TextBox id="m_txtContent" TextMode="multiline" Columns="50"
Rows="5" placeholder="Enter your comment here ..." runat="server" /></p>
        <asp:Button ID="m_btnSend" runat="server" OnClick="Button1_Click"
Text="Send" />
        <asp:Label ID="m_lblSendStatus" runat="server" Text=""></asp:Label>
    </form>
</body>
</html>
```

2) Event Handler for sending email

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Mail;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Windows.Forms;
```

```
namespace TestWeb
{
    public partial class Contact_Us : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            //Send the Email
            m_lblSendStatus.Text = "";

            string strFromEmail = m_txtFrom.Text;
            if (string.IsNullOrEmpty(strFromEmail) ||
string.IsNullOrWhiteSpace(strFromEmail))
            {
                m_lblSendStatus.ForeColor = System.Drawing.Color.Red;
                m_lblSendStatus.Text = "From address cannot be empty!";

                return;
            }

            string strContent = m_txtContent.Text;
            MailMessage mailMsg = new MailMessage();

            //Setting From , To, Sender
            mailMsg.From = new MailAddress(strFromEmail, "Jercy Liu");
            mailMsg.Subject = "Realizing Value from IT Investment";
            mailMsg.Sender = new MailAddress("web@reasonables.com");
            mailMsg.To.Add(new MailAddress("hr@reasonables.com"));
            mailMsg.Body = strContent;

            SmtpClient smtpClient = new SmtpClient("localhost", 25);
            smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;
```

```
            //smtpClient.Credentials = new System.Net.NetworkCredential("admin",
"password");
            //smtpClient.UseDefaultCredentials = true; //For using logged user
to be authenticated
            //smtpClient.EnableSsl = true; //Use SSL to send email

            try
            {
                smtpClient.Send(mailMsg);
                m_lblSendStatus.ForeColor = System.Drawing.Color.Green;
                m_lblSendStatus.Text = "Send successfully";
            }
            catch (SmtpException smtpException)
            {
                m_lblSendStatus.ForeColor = System.Drawing.Color.Red;
                m_lblSendStatus.Text = "Send failed!";
                //MessageBox.Show("Failed to send this Email: " +
smtpException.ToString());
            }
            catch (Exception exception)
            {
                m_lblSendStatus.ForeColor = System.Drawing.Color.Red;
                m_lblSendStatus.Text = "Send failed!";
                //MessageBox.Show("Unknow exception caught.");
            }

        }
    }
}
```


localhost:54381
localhost:54381
Apps
Life
Study
immig
Employment
Latest World Headlin
YouTube
FROM
jercy_lew@yahoo.com
Enter your comment here ...
Send

Q3. Write a Windows form application or iOS/Xamarin/Android that returns the difference of two given files (up to 1GB) using Levenshtein distance (edit distance).

A commonly-used bottom-up dynamic programming algorithm for computing the Levenshtein distance involves the use of an $(n + 1) \times (m + 1)$ matrix, where n and m are the lengths of the two strings. Here is pseudocode for a function *LevenshteinDistance* that takes two strings, s of length m, and t of length n, and computes the Levenshtein distance between them:

```
int LevenshteinDistance(char s[1..m], char t[1..n])
   // d is a table with m+1 rows and n+1 columns
   declare int d[0..m, 0..n]

   for i from 0 to m
       d[i, 0] := i
   for j from 1 to n
       d[0, j] := j

   for i from 1 to m
       for j from 1 to n
           if s[i] = t[j] then cost := 0
                          else cost := 1
           d[i, j] := minimum(
                                d[i-1, j] + 1,     // deletion
                                d[i, j-1] + 1,     // insertion
                                d[i-1, j-1] + cost   // substitution
                            )

   return d[m, n]
```

Two examples of the resulting matrix (the minimum steps to be taken are highlighted):

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

The invariant maintained throughout the algorithm is that we can transform the initial segment s[1..i] into t[1..j] using a minimum of d[i,j] operations. At the end, the bottom-right element of the array contains the answer.

A3

1) Main window Design source: MainWindow.Designer.cs

```
namespace FileDistance
{
    partial class MainWindow
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.m_richTxtCompareResult = new
System.Windows.Forms.RichTextBox();
            this.m_btnChooseSourceFile = new System.Windows.Forms.Button();
            this.m_txtSourceFile = new System.Windows.Forms.TextBox();
            this.m_txtTargetFile = new System.Windows.Forms.TextBox();
```

```
            this.m_btnChooseTargetFile = new System.Windows.Forms.Button();
            this.m_btnCompare = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // m_richTxtCompareResult
            //
            this.m_richTxtCompareResult.Location = new System.Drawing.Point(29,
31);
            this.m_richTxtCompareResult.Name = "m_richTxtCompareResult";
            this.m_richTxtCompareResult.ReadOnly = true;
            this.m_richTxtCompareResult.Size = new System.Drawing.Size(817,
451);
            this.m_richTxtCompareResult.TabIndex = 0;
            this.m_richTxtCompareResult.Text = "";
            //
            // m_btnChooseSourceFile
            //
            this.m_btnChooseSourceFile.Location = new System.Drawing.Point(223,
505);
            this.m_btnChooseSourceFile.Name = "m_btnChooseSourceFile";
            this.m_btnChooseSourceFile.Size = new System.Drawing.Size(132, 23);
            this.m_btnChooseSourceFile.TabIndex = 1;
            this.m_btnChooseSourceFile.Text = "Choose source file";
            this.m_btnChooseSourceFile.UseVisualStyleBackColor = true;
            this.m_btnChooseSourceFile.Click += new
System.EventHandler(this.m_btnChooseSourceFile_Click);
            //
            // m_txtSourceFile
            //
            this.m_txtSourceFile.Location = new System.Drawing.Point(29, 505);
            this.m_txtSourceFile.Name = "m_txtSourceFile";
            this.m_txtSourceFile.Size = new System.Drawing.Size(185, 21);
            this.m_txtSourceFile.TabIndex = 2;
            //
            // m_txtTargetFile
            //
            this.m_txtTargetFile.Location = new System.Drawing.Point(386, 505);
            this.m_txtTargetFile.Name = "m_txtTargetFile";
```

```
            this.m_txtTargetFile.Size = new System.Drawing.Size(171, 21);
            this.m_txtTargetFile.TabIndex = 3;
            //
            // m_btnChooseTargetFile
            //
            this.m_btnChooseTargetFile.Location = new System.Drawing.Point(564,
505);
            this.m_btnChooseTargetFile.Name = "m_btnChooseTargetFile";
            this.m_btnChooseTargetFile.Size = new System.Drawing.Size(135, 23);
            this.m_btnChooseTargetFile.TabIndex = 4;
            this.m_btnChooseTargetFile.Text = "Choose target file";
            this.m_btnChooseTargetFile.UseVisualStyleBackColor = true;
            this.m_btnChooseTargetFile.Click += new
System.EventHandler(this.m_btnChooseTargetFile_Click);
            //
            // m_btnCompare
            //
            this.m_btnCompare.Location = new System.Drawing.Point(760, 505);
            this.m_btnCompare.Name = "m_btnCompare";
            this.m_btnCompare.Size = new System.Drawing.Size(75, 23);
            this.m_btnCompare.TabIndex = 5;
            this.m_btnCompare.Text = "Compare";
            this.m_btnCompare.UseVisualStyleBackColor = true;
            this.m_btnCompare.Click += new
System.EventHandler(this.m_btnCompare_Click);
            //
            // MainWindow
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(858, 540);
            this.Controls.Add(this.m_btnCompare);
            this.Controls.Add(this.m_btnChooseTargetFile);
            this.Controls.Add(this.m_txtTargetFile);
            this.Controls.Add(this.m_txtSourceFile);
            this.Controls.Add(this.m_btnChooseSourceFile);
            this.Controls.Add(this.m_richTxtCompareResult);
            this.Name = "MainWindow";
```

```
            this.Text = "File Difference";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.RichTextBox m_richTxtCompareResult;
        private System.Windows.Forms.Button m_btnChooseSourceFile;
        private System.Windows.Forms.TextBox m_txtSourceFile;
        private System.Windows.Forms.TextBox m_txtTargetFile;
        private System.Windows.Forms.Button m_btnChooseTargetFile;
        private System.Windows.Forms.Button m_btnCompare;
    }
}
```

2) Event Handling(Including distance calculation): MainWindow.cs

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

namespace FileDistance
{
    public partial class MainWindow : Form
    {
        private string m_strSourceFile;
        private string m_strTargetFile;

        public MainWindow()
        {
            InitializeComponent();
```

```
        }

        private void m_btnChooseSourceFile_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            m_strSourceFile = "";

            if (openFileDialog.ShowDialog() == DialogResult.OK )
            {
                m_strSourceFile = openFileDialog.FileName;
                m_txtSourceFile.Text = m_strSourceFile;
            }
        }

        private void m_btnChooseTargetFile_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            m_strTargetFile = "";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                m_strTargetFile = openFileDialog.FileName;
                m_txtTargetFile.Text = m_strTargetFile;
            }
        }

        //Refer to https://www.quickdiff.com/
        private void m_btnCompare_Click(object sender, EventArgs e)
        {
            m_richTxtCompareResult.Text = string.Format("Difference Source: {0}
Target: {1}¥n",
                m_strSourceFile, m_strTargetFile);

            //m_richTxtCompareResult.Font = new Font("Consolas", 18f,
FontStyle.Bold);
            m_richTxtCompareResult.BackColor = Color.AliceBlue;

            //Read text fie
```

```
int counter = 0;
int nDistance;
string lineSrc;
string lineTarget;

// Read the file and display it line by line.
System.IO.StreamReader fileSrc =
    new System.IO.StreamReader(m_strSourceFile);
System.IO.StreamReader fileTarget =
    new System.IO.StreamReader(m_strTargetFile);
while ((lineSrc = fileSrc.ReadLine()) != null &&
    (lineTarget = fileTarget.ReadLine()) != null)
{
    lineSrc = lineSrc.Trim();
    lineTarget = lineTarget.Trim();

    nDistance = LevenshteinDistance(lineSrc, lineTarget);
    if (nDistance != 0)
    {
        m_richTxtCompareResult.SelectionBackColor = Color.Plum;
        m_richTxtCompareResult.AppendText("-" + lineSrc);
        m_richTxtCompareResult.SelectionBackColor = Color.AliceBlue;
        m_richTxtCompareResult.AppendText("¥n");

        m_richTxtCompareResult.SelectionBackColor = Color.Aquamarine;
        m_richTxtCompareResult.AppendText("+" + lineTarget);
        m_richTxtCompareResult.SelectionBackColor = Color.AliceBlue;
        m_richTxtCompareResult.AppendText("¥n");
    }
    else
    {
        m_richTxtCompareResult.SelectionBackColor = Color.AliceBlue;
        m_richTxtCompareResult.AppendText(lineSrc);
        m_richTxtCompareResult.SelectionBackColor = Color.AliceBlue;
        m_richTxtCompareResult.AppendText("¥n");
    }
    counter++;
}
```

```
            //Remaining lines
            while((lineSrc = fileSrc.ReadLine()) != null)
            {
                m_richTxtCompareResult.SelectionBackColor = Color.Plum;
                m_richTxtCompareResult.AppendText("-" + lineSrc);
                m_richTxtCompareResult.SelectionBackColor = Color.AliceBlue;
                m_richTxtCompareResult.AppendText("¥n");
            }

            while ((lineTarget = fileTarget.ReadLine()) != null)
            {
                m_richTxtCompareResult.SelectionBackColor = Color.Aquamarine;
                m_richTxtCompareResult.AppendText("+" + lineTarget);
                m_richTxtCompareResult.SelectionBackColor = Color.AliceBlue;
                m_richTxtCompareResult.AppendText("¥n");
            }

            fileSrc.Close();
            fileTarget.Close();
        }

        // Calculate Levenshtein Distance
        //
https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistan
ce/Levenshtein%20Distance.htm
        private int LevenshteinDistance(string s, string t)
        {
            int m = s.Length;
            int n = t.Length;

            if (m == 0)
            {
                return n;
            }

            if (n == 0)
            {
```

```
            return m;
        }

        int cost;

        int[,] d = new int[m, n];

        for (int i = 0;i < m;i++)
        {
            d[i, 0] = i;
        }

        for (int j = 1; j < n; j++)
        {
            d[0, j] = j;
        }

        for (int i = 1; i < m;i++)
        {
            for (int j = 1; j < n;j++)
            {
                cost = (s[i] == t[j]) ? 0 : 1;

                //d[i - 1, j] + 1,     // deletion
                //d[i, j - 1] + 1,     // insertion
                //d[i - 1, j - 1] + cost   // substitution
                d[i, j] = Math.Min(d[i - 1, j] + 1, d[i, j - 1] + 1);
                d[i, j] = Math.Min(d[i, j], d[i - 1, j - 1] + cost);
            }
        }
        // Console.Write(string.Format("Distace: %d, %d deletions, %d
insertions and %d substitution"));
        return d[m-1, n-1];
    }
  }
}
```

3) Testing

Test file:

file1.txt file2.txt