DATABASE TERMNOLOGY

ADO .Net (ActiveX Data Objects) provides an API (application programming interface) for accessing database systems programmatically.

ADO .Net is designed to interact with Microsoft’s component object mode (com) framework.

ADO .Net Namespaces:

1. System.Data.System.Data.OleDb

2. System.Data.SqlClient

Namespace System.Data is the root namespace for the ADO .Net APL.

System.Data.OleDb and System.Data.SqlClient contains classes that enable programs to connect with and modify datasources.

System.Data.OleDb has classes designed to work with any datasource.

System.Data.SqlClient has classes to connect that are optimized to work with Microsoft SqL Server 2000 databases.

Instances of System.Data.DataSet consist of a set of DataTables and relationships among those DataTables represent a cache of data (data that a program stores temporarily in local memory).

Structure of a DataSet mimics the structure of a relational database.

DataSet is disconnected which means the program does not read a persistent connection to the datasource to work with data in a DataSet.

The program connects to the datasource only during the initial population of the DataSet initially and then to store any changes made in the DataSet.

The program does not require any active, permanent connection to the datasource.

Instances of class OleDbConnection of namespace System.Data.OleDb represent a connection to a datasource.

Instances of class OleDbDataAdapter connect to a datasource through an instance of class OleDBConnection and can populate DataSets with data from a datasource.

Instances of class OleDBCommand of namespace System.Data.OleDb represent an arbitrary SQL command to be executed on a datasource.

A program can use instances of class OleDbCommand to manipulate a datasource through an OleDbConnection.

The programmer must close the active connection to the datasource explicitly once no further changes are to be made.

Note: OleDbCommand objects do not cache data in local memory.

DataGrid (a component from namespace System.Windows.Forms) that can display a datasource GUI.

You can register a database as a datasource by : (1) using ALT + V + V to display the Server Explorer and click Data Connections node; (2) Click Add Connection in Provider tab choose Microsoft Jet 4.0 OLEDB Provider, which is the driver for Access databases; (3) In Connectioin tab select the location of the desired database in Select Access Database.

This creates an OleDbConnection to the source, which the Windows Form Designer shows as OleDbConnection1. (4) Drag an OleDbdataAdapter from the toolbox’s Data subheading onto the Windows Form Designer; (5) Data Adapter Configuration Wizard is displayed which configures the OleDbDataAdapter instance with a custom query for populating a DataSet. (a) Next displays a drop-down list of possible connections. Select the connection created in the previous step from drop-down list. (b) Next screen allows you to choose how the OleDbAdapter should access the database. Keep the default Use SQL Statement option. (c) Next click Query Builder button, select the desired table from Add menu, then Close menu. Place a check mark in the “ \* All Columns” box from the small Table window (should list all fields of the table). Click Ok and Finish; (6) Drag DataSet from Data tab which displays the Add DataSet window. Choose Untyped Dataset (no schema) since the query which we populate the DataSet dictates the DataSet schema or structure. (Tables that comprise DataSet and the relationship among those tables); (7) finally, add the DataGrid.

Using Code

Dim dt As New DataTable()

Dim ConnStr As String = “Provider = Microsoft.Jet.OLEDB.4.0.;” & “Data Source = Database\_Name.MDB”

Dim dataAdapter As New OleDb.OleDbDataAdapter(query, connStr)

dataAdapter.Fill (dt)

dataAdapter.Dispose()

DataGrid.DataSource = dt

Note: DataGrid can be used to add, modify, and delete records from a database.

Changing the Contents of a Database

Dim commandBuilder As New OleDbCommandBuilder (dataAdapter)

Note: This will automatically enable any modifications to be stored.

If changes is an Integer variable Then

Changes=DataAdapter.Update(dt) will store all of the instructions, updates, and deletions made in the data table to the database and assign the number of records changed to the variable Changes.

Note: dt.Clear clears the current contents of the table.

dt.Rows.Count is the number of records in the table.

The value of dt.Columns.Count is the number of columns in the table.

The records are numbered 0 through dt.Rows.Count -1.

The fields are numbered 0 though dt.Columns.Count -1.

The value of dt.Columns(j) is the name of the jth field.

The value of dt.Rows(i)(j) is the entry in the jth field of the ith record.

The value of dt.Rwos(i) (fieldName) is the string containing the entry in the specified field of the ith record.

Values from Cities table of Megacities.mdb

Expression Value

dt.Rows.Count 10

dt.Columns.Count 4

dt.Rows(3)(1) USA

dt.Rows(2)(“City”) Calcutta

ASP.NET WEBPAGES

Web Programming using Visual Basic .Net and ASP

History of Web Programming:

HTML has provided a way for users to input information into a Web page using what is called an HTML form. By providing a supplementary type of program named Common Gateway Interface (CGI), Web sites could receive and process information entered into HTML forms. CGI programs are still used, but must be carefully designed to prevent intrusions by hackers.

Web pages can also contain short programs named scripts. Scripts are usually written in languages such as VBScript, JavaScript, Perl, PHP, or Python. A server-side script is executed by the client Browser, so it does not require any processing by the Web server. Client-side scripts are often used to perform validation of user input, fancy animations on Web pages, and in general to simulate the kind of user interfaces found in Windows Desktop applications. Scripting languages are tricky to use because they do not perform compile-time type checking. As a result, runtime errors occur at unpredictable moments, confusing end users.

The technology for processing Web pages and creating Web applications has evolved quite a bit during the past several years. ASP.NET and competing products make it easy to create Web applications as powerful as desktop applications. Best of all, the power of object-oriented programming, with features such as runtime exception handling, inheritance, and database classes have become available to Web programmers.

What is ASP.NET

ASP.NET is called a platform because it provides development tools, code libraries, and visual controls for Web programming. It contains Web-friendly counterparts of the Windows Forms classes that we have already using. You can use ASP.NET from Visual Basic or C++ because both languages share the same tools, libraries, and controls. The essential elements of the ASP.NET platform are:

Syatem.Web namespaces, containing classes specifically designed for Web programming.

Web forms controls (also known as server controls)

Visual Basic and C++ programming languages

The .Net framework (integrated collection of classes and methods)

ADO.Net database classes

ASP.NET gives end users the illusion that Web applications behave almost like desktop applications. The programmer, on the other hand, is given tools that assist in creating visual interfaces and in writing object-oriented code. ASP.NET lets programmers use object-oriented classes and interactive controls similar to those used in desktop applications. Visual Studio .NET is an excellent integrated development environment for building and test .NET applications.

Creating ASP.NET Web Applications

Web applications written for ASP.NET consist of several parts:

Content: Web forms, HTML code, Web forms controls, images, and other multimedia

Program logic: both scripted and compiled executable code

Configuration information

The computer running a Web application must be running a Web server such as Internet Information Services (IIS). Windows XP and Windows 2000 include IIS as an optional service. When you create a Web application in ASP.NET, you usually start with a Web page having a special filename extension: aspx. This page is commonly known as a Web form because it contains an HTML form and HTML based controls such as buttons and text boxes. HTML controls and Web forms controls are added to the page using an ordinary text editor or an integrated tool such as Visual Studio .NET. This part of the application is called content.

Web forms controls are interactive controls such as buttons, list boxes, and text boxes that execute on the server. Though they look like HTML controls, they are more powerful because they can use event handler procedures to carry out actions based on user input. In effect, they behave a lot like Windows desktop controls. The source code related to a Web form is stored in a related file called a codebehind file, with the filename extension aspx.vb. This part of the application is called the program logic.

Configuration information is stored in two files: (1) Web.config describes the runtime environment. For example, you might use it to specify a connection to a database; (2) Styles.css is a Cascading Style Sheet (CSS) file that contains HTML styles for customizing the appearance of Web forms. (Most Web designers use cascading style sheets for ordinary Web pages)

A Web application can contain any number of Web forms, each of which is assigned a codebehind file. When the program is compiled, the codebehind file is translated into a Dynamic Link Library (DLL) file containing compiled Intermediate Language (IL) code. When the program runs, the IL code is converted to native machine code by the Microsoft Just-In-Time (JIT) compiler. A Web application executes when the URL of its startup Web form is requested by a Web browser. The Web server interprets the contents of the startup form. An application can run on localhost, or it can be run over a network (including the Internet) by users connected to the Web server.

HTML Designer

The HTML Designer is a tool supplied inside Visual Studio .NET that simplifies the design of ASP.NET pages. It generates HTML source code and embeds the ASP.NET tags required for Web forms controls. You do not have to use the HTML designer. Some prominent Web developers prefer to use a plain text editor. The Visual Studio .Net ToolBox has two tabs that identify groups of controls:

The HTML group contains standard HTML controls that might be found on any Web page. They are 100-percent compatible with standard HTML, and as such have a limited number of properties, not event handling, and no associated classes. Any HTML control can be interpreted and translated into HTML by the Web server. In any event, such a control would still not have the rich set of properties found in a corresponding Web forms control.

The Web Forms group contains Web forms controls that rung on a Web server. These easy-to-use controls have a rich set of properties and supporting classes. Browser do not understand the various properties found in Web forms controls, so the Web server must interpret each control and translate it into standard HTML. Because of their power and flexibility, Web forms controls are used most of the time when writing ASP.NET applications.

Comparing TextField to TextBox

The TextField is an HTML control that has seven properties. The TextBox is a Web forms controls and has twenty-one properties. A Web designer who uses only HTML would have to go to great lengths to simulate the richness of the TextBox control using client-side scripts. The difference between TextField control and TextBox control are typical of the differences between other HTML and Web forms controls.

Form Layout Options

Grid Layout is the default. Controls can be positioned on a grid by dragging them with the mounts or yr pressing keyboard arrow keys.

Flow Layout is what HTML developers use for Web pages. Controls appear on the form in sequential order, much like the words in a text document. You can use HTML tables to position controls anywhere on the Web page.

To select the layout method, click the mouse in an open area of a new Web form, and select the page layout property.

Files in Web Projects

Web Form consists of two files having filename extensions aspx and aspx.vb. The aspx file contains the visible HTML code, and the aspx.vb file is the codebehind file that supports the Web page with program code. When the latter file is compiled, its code is stored in a DLL.

ASP.NET Application file is named Global.asax and contains program code that manipulates Application and Session objects:

An Application object contains properties and methods relating to the ASP.NET application as a whole. Application data are shared between all client connections active at a given moment.

A Session object contains properties and methods relating to an individual client connection (called a session). The client can, for example, create session variables that pertain only to a single end user.

AssemblyInfo.vb contains project information in a text file that can be configured by the developer. It contains information such as the company name, copyright, and program version number.

ProjectName.vsdisco is a XML discovery file that holds links to required Web services. Its filename consists of the project name followed by the filename extension vsdisco.

Styles.css is a cascading style sheet file containing definitions of paragraph and font styles. You can use style sheets to redefine existing HTML tags such as H1 (heading level 1) or P (paragraph). You can also define new style tags that customize your program’s appearance.

Web.config (Web configuration file) contains information about each URL resource used in the project. For example, you can use it to customize error messages displayed to the user, set the authentication policy of the program, or store a database connection string.

DLL file is a compiled code created the first time an ASP.NET application runs. The file has the same name as the project, along with a filename extension of DLL. This file is loaded by the server when a Web application is started, and stays in memory as long as the application is running.

The Codebehind file is filename.asxp.vb

Looking at the HTML code you can easily identify Web forms controls because they start with the “<asp:” tag.

The runat=”server” attribute says the Web server will process this control before generating standard HTML for the user’s browser.

Physical File Locations

If you selected <http://localhost/Test1> as your project location, its actual physical location on your drive is in the default Web server directory. On most computers, the default Web server is located at c:\Inetpub\wwwroot. Assuming that is the case, the program created will be stored at c:\Inetpub\wwwroot\Test1. Aside from the program’s project files, Visual Studio .NET places the project’s .sln and .suo files in the default Visual Studio .NET projects folder. These two files are saved separately form the rest of the project.

HyperText Links

An absolute link contains the complete URL of the destination. An example is <http://localhost/myProg/Default.aspx>.

A relative link stores only the path from the current page to some other page. Suppose your program was located at localhost/myProg/Default.aspx, and you wanted the user to navigate to a page named Confirm.aspx in the localhost/myProg/special folder. The following relative link would accomplish the navigation: special/Confirm.aspx.

An easy way to navigate from one page to another is to use a HyperLink control, located in the HTML section of the Visual Studio .Net Toolbox. Simply assign the destination URL to the control’s NavigateUrl property. However, if you want to write an event handler that navigates between pages, there are two common ways to write it:

When you call Server.Transfer to move to a different page, pass it the relative path of the destination page. This method requires the destination page to be located on the same Web server as the current page. Suppose the file Confirm.aspx was in the same directory as the current page; then you could write: Server.Transfer(“Confirm.aspx”). One consideration should be noted: The user’s Web browser is not informed that the current page’s URL has changed, so users may get unpredictable results if they click on their browser’s Refresh button.

You can navigate to another Web page by calling Respnose.Redirect, passing it either a relative path or a complee URL. The destination can be either on the same server or on a different server. If we wanted to load The University of Alabama home page, the required statement would be: Response.Redirect([http://www.ua.edu](http://www.ua.edu/)). Response.Redirect is slightly less efficient than Server.Transfer, because if forces a roundtrip to the server (a postback). If the user clicks the Refresh button after a Response.Redirect has taken place, the browser is aware that the current page’s URL has changed.

Formatting Tips

To set the tab order on a Web form, manually set the TabIndex property of each control. The index values start at 1 rather than 0. In Internet Explorer, the first time the user presses Tab, the browser’s Address bar gets the focus; when Tab is pressed again, the focus moves to the first field in your tab sequence. Not all browsers support the TabIndex property, but you can definitely use it with Internet Explorer and later.

For proper compatibility with non-Microsoft browsers, a text box containing multiple lines should not be dragged with the mouse to increase its size. Instead, specify values for the Rows and Columns properties. The Rows property specifies the height in terms of text lines. The Columns property specifies the character width of the test box.

Grid Layout versus Flow Layout

Every page object has a property named pageLayout, which has one ot two possible values: GridLayout or FlowLayout. The default is GridLayout, perhaps to accommodate programmers who are accustomed to creating Windows forms. Grid loyout, however, is not typically used by HTML designers. There are a few disadvantages to using grid layout:

You cannot type text directly onto the form. All text must be placed in Label controls.

Grid layouts do not work if you are using HTML tables elsewhere on the form.

When the user changes the default text size setting in their browser, the contests of Label controls may wrap around and look ugly.

The form’s HTML code is complicated by having to introduce LEFT and TOP property values into each control’s definition. Example: in flow layout mode, you could insert a level-2 heading on a page with just one line of HTML: <H2>Label Controls</H2> ; but in grid layout mode, the following HTML is required which includes a property that indicates the temporary use of flow layout:

<Div stype=”Display: inline; z-INDEX: 103; LEFT: 24px;

WIDTH: 312px; POSITION: absolute; TOP: 16px; HEIGHT: 23px”;

ms\_positioning=”FlowLayout”>

<H2>Label Controls</H2>

</Div>

Wrapround text can be a problem in grid layout mode when users switch to larger fonts in their Web browser.

HTML Table Control

The HTML Table control is particularly useful when working in flow layout mode. You can use it to align text and Web controls. Normally, we avoid HTML controls because they do not generate programmable events. The Table control is an exception because it is only used for formatting. You can use Visual Studio’s Table menu command to insert, delete, and select rows and columns. The following keyboard shortcuts speed up the building of tables:

Insert a new row above the current row: Crtl + Alt + Up Arrow

Insert a new row below the current row: Crtl + Alt + Down Arrow

Insert a new column to the left of the current row: Crtl + Alt + Left Arrow

Insert a new column to the right of the current row: Crtl + Alt + Right Arrow

Although table lines show in design mode, you can make them disappear at runtime by setting the Table’s border property equal to zero. When the program runs, the alignment is exact, regardless of the text size selected by the user.

Using Tables to Design Forms

Decide on an overall organizational format for the page. Example, you might divide up the page into three large columns with only one cell in each column. In this case you would set valign (vertical alignment) property of each column to top and insert a single character to act as a place holder. This would result in a column-major layout scheme.

Insert a new HTML table into each cell of the first table. This action allows each column to be independent of the other two. Leave at least two columns in each of these tables, to make easier to provide empty space between columns A, B, and C. Also, it is easier to view the individual cells if you insert a space in each cell. As you build each column, always leave one table row between controls. You will need to use the empty rows to fine-tune the vertical spacing.

After you have inserted an ample number of rows in each tale, begin inserting text and controls in the table columns. Setting the border thicknesses of the alignment tables to zero so they will be invisible at runtime.

Row-Major Layout

Instead of dividing a page into columns, you can divide it into rows. Insert a separate table into each row. This approach works well for input fields that very in length. Labels are easily right-justified by setting the align property if their enclosing table cells to right.

Merging Cells

Merging cells together to create larger cells can be a little tricky, but is very useful when you have too much data for one cell.

Merge all the cells in a row together by first selecting the Table | Select | Row menu command, followed by the Table | Merge Cells command.

Merge all the cells in a column together by first selecting the Table | Select | Column menu command, followed by the Table | Merge Cells command.

Be cautious when using rows and columns. Doing so can have drastic effects on other rows and columns in your table. Fortunately, using a separate table for each column helps to minimize the need for row and column spans.

If the height of a row seems to change when you switch from design mode to runt mode, drag the bottom of the row with the mouse. This causes a specific row height to be encoded in the Style property of each cell in the row.

Start wit more columns and rows than you think you need. It’s much easier to delete an existing column than to insert a new one without messing up the original table alignment.

In design mode, avoid pressing the Enter key as the last action while editing a cell. Doing so inserts a paragraph tag which is difficult to remove. You can do it by editing the HTML directly, and removing the <P> and </P> tags from the cells. In HTML, a table cell is defined by the <TD> and </TD> tags.

Points for Flow Layout

Flow layout is more compatible with existing HTML design programs, making it easier for you to customize your program’s appearance with third-party Web editing tools.

Flow layout automatically adjusts to changes in the size of the text content in Web forms. Grid layout requires significant manual adjustments to accommodate such changes.

PROJECT 1

Module Module1

Sub Main()

Dim loanDecimal, loanAmount, loanPercent, monthlyPayment, interestAmount, principal, balance As Double

Dim numOfYears, period As Integer

Console.Write("Enter rate of loan in decimal form: ") 'THE DECIMAL

loanDecimal = CDbl(Console.ReadLine)

Console.Write("Enter the number of years: ") 'THE NUMBER OF YEARS

numOfYears = CInt(Console.ReadLine)

Console.Write("Enter the amount of loan: ") 'OUTPUT THE AMOUNT OF THE LOAN

loanAmount = CDbl(Console.ReadLine)

loanPercent = loanDecimal \* 100

Console.WriteLine("")

Console.WriteLine("")

Console.WriteLine("Rate of Loan= {0}%", loanPercent)

Console.WriteLine("Number of Years= {0}", numOfYears)

Console.WriteLine("Amount Borrowed= {0:C}", loanAmount)

monthlyPayment = Pmt((loanDecimal / 12), (numOfYears \* 12), (-loanAmount))

Console.WriteLine("Amount of Monthly Payment= {0:C}", monthlyPayment)

Console.WriteLine("")

Console.WriteLine("")

Console.WriteLine("Amorization Table")

Console.WriteLine("Period{0}Payment{0}Interest{0}{0}Principal{0}{0}Balance", Constants.vbTab)

balance = loanAmount

period = 1

'RUN LOOP FOR EACH PERIOD

While period <= (numOfYears \* 12)

principal = PPmt((loanDecimal / 12), period, (numOfYears \* 12), (-loanAmount))

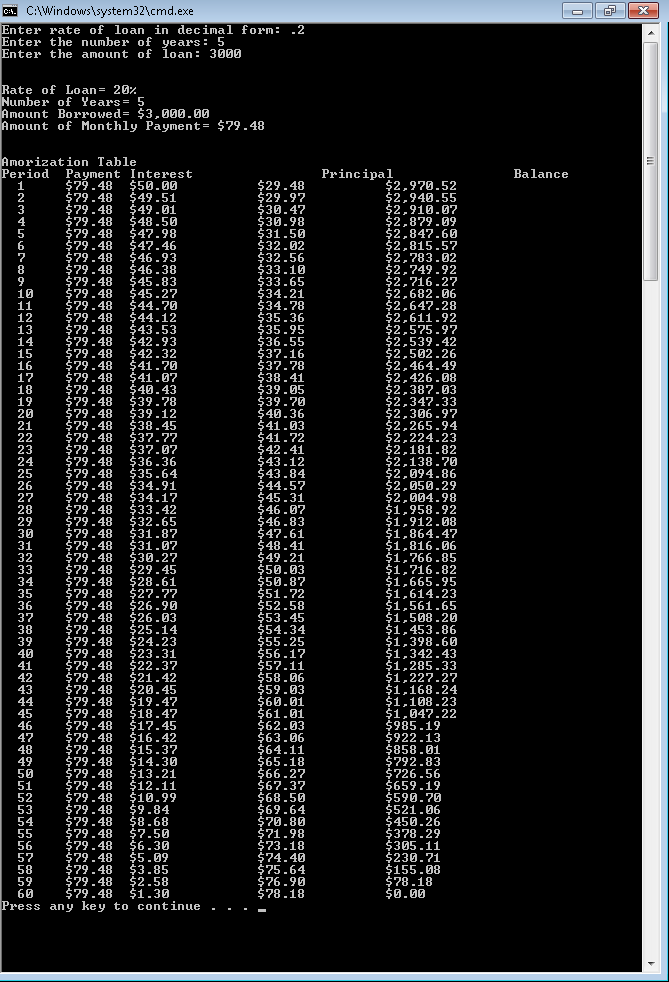
interestAmount = IPmt((loanDecimal / 12), period, (numOfYears \* 12), (-loanAmount))

'REMAINING BALANCE - MONTHLY PAYMENT AMOUNT + INTEREST AMOUNT

balance = balance - monthlyPayment + interestAmount

'OUTPUT TABLE INFORMATION

Console.WriteLine(" {1}{0}{2:C}{0}{3:C}{0}{0}{4:C}{0}{0}{5:C}", Constants.vbTab, period, monthlyPayment, interestAmount, principal, balance)

 period = period + 1

End While

End Sub

End Module

PROJECT 2

Public Class Form1

Dim Megan(2, 3, 1) As Integer

Dim sum, Money, Total\_Money, GrandTotal As Integer

Dim average, OA As Double

Dim Govmt(,,) As Integer = New Integer(2, 3, 1) {}

Dim Employee(,,) As Integer = New Integer(2, 3, 1) {}

Private Sub btnexit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnexit.Click

Me.Close()

End Sub

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

lblDate.Text = DateTime.Now.ToLongDateString()

lbltime.Text = DateTime.Now.ToLongTimeString()

End Sub

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Call Load\_Array(Megan)

End Sub

Private Sub Load\_Array(ByVal Megan(,,) As Integer)

Dim Div1, Stp1, LH1, Div2, Stp2, LH2 As Integer

Dim Input As IO.StreamReader = IO.File.OpenText("Government.txt")

For LH1 = 0 To 1

For Div1 = 0 To 2

For Stp1 = 0 To 3

Govmt(Div1, Stp1, LH1) = CInt(Input.ReadLine)

Next Stp1

Next Div1

Next LH1

Input.Close()

Dim Input2 As IO.StreamReader = IO.File.OpenText("Employee.txt")

For LH2 = 0 To 1

For Div2 = 0 To 2

For Stp2 = 0 To 3

Employee(Div2, Stp2, LH2) = CInt(Input2.ReadLine)

Next Stp2

Next Div2

Next LH2

Input2.Close()

End Sub

Private Sub results\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles results.Click

Dim k, m, r As Integer

For k = 0 To 1

If k = 0 Then

LstPayment.Items.Add("Low =1")

Else

LstPayment.Items.Add("High =2")

End If

For m = 0 To 2

LstPayment.Items.Add("GS-" & m + 1 & "=" & m + 1)

For r = 0 To 3

LstPayment.Items.Add("STEP =" & r + 1 & " " & Format$(Govmt(m, r, k), "currency"))

Next

Next m

Next k

Dim LH1, Div1, Stp1, q As Integer

Dim s, num(6), avge(2) As Integer

LstResults.Items.Add("The Results of Federal Government Pay Calculations are:")

q = 0

For LH1 = 0 To 1

If LH1 = 1 Then

q = 1

End If

For Div1 = 0 To 2

Total\_Money = 0

average = 0

For Stp1 = 0 To 3

Money = Govmt(Div1, Stp1, LH1) \* Employee(Div1, Stp1, LH1)

Total\_Money = Total\_Money + Money

average = average + Employee(Div1, Stp1, LH1)

avge(q) = avge(q) + Employee(Div1, Stp1, LH1)

If Stp1 = 3 Then

num(s) = Total\_Money

s = s + 1

LstResults.Items.Add("Sum for Gs-" & s & "is" & " " & Format$(Total\_Money, "currency") & vbTab & "The Average is" & Format$(Total\_Money \ average, "currency"))

End If

Next Stp1

Next Div1

Next LH1

LstResults.Items.Add("[][]")

q = 0

For s = 0 To 3 Step 3

sum = num(s) + num(s + 1) + num(s + 2)

GrandTotal = GrandTotal + sum

If s = 0 Then

LstResults.Items.Add("Low Sum = " & Format$(sum, "currency") & vbTab & "Low Average = " & Format$(sum \ avge(q), "currency"))

Else

LstResults.Items.Add("High Sum = " & Format$(sum, "currency") & vbTab & "High Average = " & Format$(sum \ avge(q + 1), "currency"))

End If

Next

LstResults.Items.Add("Grand Total = " & Format$(GrandTotal, "currency") & vbTab & "Overall Average = " & Format$(GrandTotal \ (avge(q) + avge(q + 1)), "currency"))

End Sub

Private Sub clear\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles clear.Click

LstPayment.Items.Clear()

LstResults.Items.Clear()

End Sub

Private Sub wite\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles wite.Click

Dim Output As IO.StreamWriter = IO.File.CreateText("Low\_Data.txt")

Dim Output2 As IO.StreamWriter = IO.File.CreateText("High\_Data.txt")

Dim a, b, c As Integer

For c = 0 To 1

For a = 0 To 2

For b = 0 To 3

If c = 0 Then

Output.WriteLine(Format$(Govmt(a, b, c), "currency" & " " & Employee(a, b, c)))

Else

Output2.WriteLine(Format$(Govmt(a, b, c), "currency" & " " & Employee(a, b, c)))

End If

Next

Next

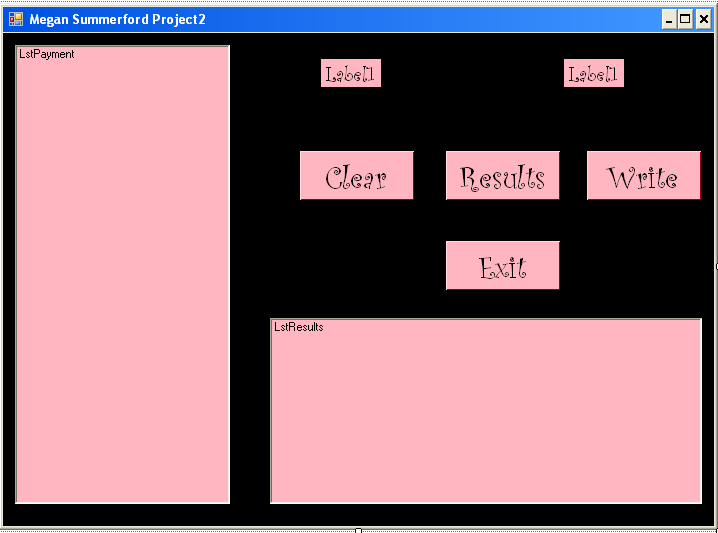
Next

Output.Close()

Output2.Close()

End Sub

End Class



PROJECT 3

Public Class Form1

Dim isSimRunning As Boolean = False

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

timeLabel.Text = DateTime.Now.ToLongTimeString() 'UPDATES TIME WITH CURRENT TIME

dateLabel.Text = DateTime.Now.ToLongDateString() 'UPDATES DATE WITH CURRENT TIME

End Sub

Private Sub exitButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles exitButton.Click

Me.Close() 'EXITS PROGRAM

End Sub

Private Sub clearButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles clearButton.Click

simulationTextbox.Clear() 'CLEARS SIMULATION TEXT BOX

gallonsTextbox.Clear() 'CLEARS GALLONS TEXT BOX

priceTextbox.Clear() 'CLEARS PRICE TEXT BOX

statusListBox.Items.Clear() 'CLEARS STATUS LIST BOX

isSimRunning = False 'SETS SIMULATION RUNNING STATUS TO FALSE

End Sub

Private Sub simButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles simButton.Click

If (simulationTextbox.TextLength > 0) Then

'PREVENTS FROM SIMULATING WITH NO SIMULATION TIME ENTERED

Dim enteredSimTime As Integer = CInt(simulationTextbox.Text)

'CONVERTS SIMULATION TIME TO INTEGER

If ((enteredSimTime >= 100) And (isSimRunning = False)) Then

'IF SIMULATION TIME IS LESS THAN 100 AND IS NOT RUNNING, THEN DISPLAY ERROR MESSAGE

isSimRunning = True 'TO PREVENT MULTIPLE INSTANCES OF SIMULATION

Dim pumpClass As New Pump() 'CREATES NEW PUMP OBJECT

gallonsTextbox.Text = pumpClass.galProp() 'OUTPUTS INITIAL GALLONS IN TANK

priceTextbox.Text = pumpClass.priceProp().ToString("c", New Globalization.CultureInfo("en-US", False))

'OUTPUTS PRICE OF GALLON OF GAS

Dim totalElapsedTime As Integer = 0 'SETS ELAPSED TIME TO ZERO

statusListBox.Items.Add("Starting a new simulation...")

statusListBox.Items.Add("Simulation Time is " + CStr(enteredSimTime) + " minutes.")

While (enteredSimTime >= totalElapsedTime)

Dim customerClass As New Customer() 'CREATES NEW CUSTOMER OBJECT

Dim randIdle As Integer = customerClass.randomIdleTimeProp()

'PUTS RANDOM IDLE TIME INTO VARIABLE

Dim randGas As Integer = customerClass.randomGasPumpedProp()

'PUTS RANDOM GALLONS PUMPED INTO VARIABLE

statusListBox.Items.Add(pumpClass.displayPumpTankStats())

'DISPLAY NUMBER OF GALLONS OF GAS IN TANK

statusListBox.Items.Add(pumpClass.displayPumpPriceStats())

'DISPLAY PRICE OF GAS PER GALLON

totalElapsedTime = randIdle + totalElapsedTime

'ADD RANDOM ELAPSED TIME TO TOTAL ELAPSED TIME

statusListBox.Items.Add("The idle time is " + CStr(randIdle) + \_

" minutes and we are " + CStr(totalElapsedTime) + " minutes into the simulation.")

'DIPSLAY IDLE TIME AND TOTAL TIME

statusListBox.Items.Add(pumpClass.pumpTransaction(randGas))

'DISPLAY GAS PUMP TRANSACTION

End While

isSimRunning = False 'END OF SIMULATION, CHANGE STATUS TO REFLECT

statusListBox.Items.Add("Time is up. The simluation has ended.") 'NOTICE OF END OF SIMULATION

Else

'OUTPUTS ERROR MESSAGE IF SIMULATION TIME IS NOT GREAT ENOUGH

statusListBox.Items.Add("Simulation Time needs to be equal or greater than 100 minutes")

isSimRunning = False

End If

Else

statusListBox.Items.Add("Simulation Time was not entered. Please enter a time.")

'ERROR MESSAGE

End If

End Sub

Public Class Pump

Public gal As Integer 'GALLONS AVAILABLE OF GAS

Private price As Decimal 'PRICE OF GAS

Private tankLevel As Integer 'TANK LEVEL OF GAS

Dim gasPumped As Integer 'AMOUNT OF GAS PUMPED

Dim pricePumped As Double 'PRICE OF THE AMOUNT OF GAS PUMPED

Public Sub New() 'DEFAULT CONSTRUCTOR

gal = 500 'INITIALIZE GALLONS TO 500

price = 2.9 'INITIALIZE PRICE TO 2.90

End Sub

Public Property priceProp() As Decimal

'GETTER AND SETTER FOR PRICE

Get

Return price

End Get

Set(ByVal value As Decimal)

If value >= 0 Then

price = value

End If

End Set

End Property

Public Property galProp() As Integer

'GETTER AND SETTER FOR GALLONS

Get

Return gal

End Get

Set(ByVal value As Integer)

If gal >= 0 Then

gal = value

End If

End Set

End Property

Public Property tankLevelProp() As Integer

'GETTER AND SETTER FOR TANK LEVEL

Get

Return tankLevel

End Get

Set(ByVal value As Integer)

If value >= 0 Then

tankLevel = value

End If

End Set

End Property

Public Function displayPumpTankStats() As String

Return "The gas tank has " + CStr(galProp()) + " gallons of gas."

'DISPLAYS THE AMOUNT OF GAS IN THE TANK

End Function 'END OF DISPLAYPUMPSTATS FUNCTION

Public Function displayPumpPriceStats() As String

Return "The price per gallon of gas is " + CStr(priceProp().ToString("c", New Globalization.CultureInfo("en-US", False)))

'DISPLAYS THE PRICE PER GALLON OF GAS

End Function

Public Function pumpTransaction(ByVal gasRequested As Integer) As String

Dim finalOutputString As String

If gasRequested > gal Then

'IF THERE IS MORE GAS REQUESTED THAN WHAT IS AVAILABLE

gasPumped = gal

Else

gasPumped = gasRequested

End If

gal = gal - gasPumped 'CALCULATE THE AMOUNT OF GAS LEFT IN TANK

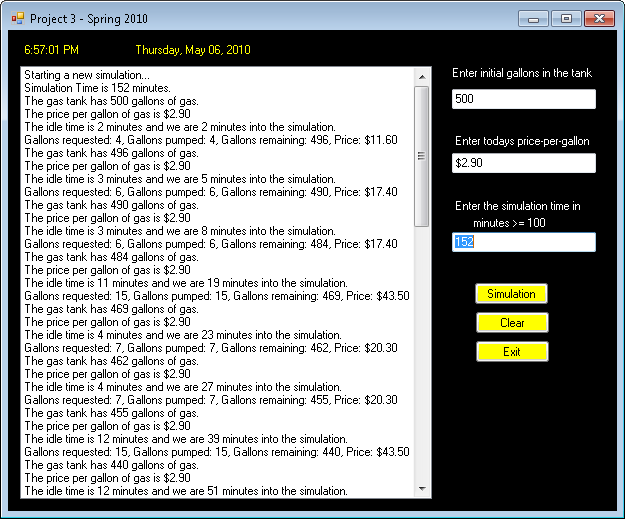
pricePumped = gasPumped \* price 'CALCULATE PRICE OF GAS THAT WAS PUMPED

'DISPLAY OUTPUT

finalOutputString = "Gallons requested: " + CStr(gasRequested) + ", Gallons pumped: " + CStr(gasPumped) + \_

", Gallons remaining: " + CStr(gal) + ", Price: " + CStr(pricePumped.ToString("c", New Globalization.CultureInfo("en-US", False)))

Return finalOutputString

 End Function 'END OF PUMPTRANSACTION FUNCTION

End Class 'END OF PUMP CLASS

Public Class Customer

Private randomIdleTime As Integer

Private randomGasPumped As Integer

Public Sub New()

randomIdleTime = randNumber(1, 16) 'RANDOM AMOUNT OF IDLE TIME

randomGasPumped = randNumber(3, 20) 'RANDOM AMOUNT OF GAS PUMPED

End Sub

Public Property randomIdleTimeProp() As Integer

'GETTER AND SETTER FOR RANDOM IDLE TIME

Get

Return randomIdleTime

End Get

Set(ByVal value As Integer)

value = randNumber(1, 16)

If value >= 0 Then

randomIdleTime = value

End If

End Set

End Property

Public Property randomGasPumpedProp() As Integer

'GETTER AND SETTER FOR RANDOM AMOUNT OF GAS PUMPED

Get

Return randomGasPumped

End Get

Set(ByVal value As Integer)

value = randNumber(3, 20)

If value >= 0 Then

randomGasPumped = value

End If

End Set

End Property

Public Function randNumber(ByVal maxOfRange As Integer, ByVal minOfRange As Integer) As Integer

'START RANDOM NUMBER GENERATOR

Dim r As New Random(System.DateTime.Now.Millisecond)

If minOfRange > maxOfRange Then 'IF THE MIN NUMBER IS OUTSIDE OF THE RANGE

Dim t As Integer = minOfRange 'COPY NUMBER TO TEMP VARIABLE

minOfRange = maxOfRange 'SET MAX AS MIN

maxOfRange = t 'SET TEMP TO MAX

End If

Return r.Next(minOfRange, maxOfRange) 'RETURNS RANDOM VALUE

End Function 'END OF RANDNUMBER FUNCTION

Protected Overrides Sub Finalize() 'DESTRUCTOR

End Sub

End Class 'END OF CUSTOMER CLASS

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

End Sub

End Class 'END OF FORM1 CLASS

PROJECT 4

Imports System.IO

Imports System.Globalization

Public Class project4

Dim totalPurchasePrice As Decimal 'TOTAL PRICE THAT INCLUDE

Dim taxRate As Decimal = 0.09 'TAX RATE / PERCENTAGES TAX

Dim subtotalPurchasePrice As Decimal 'SUBTOTAL PRICE

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

dateLabel.Text = DateAndTime.Now.ToLongDateString()

timeLabel.Text = DateAndTime.Now.ToLongTimeString()

End Sub

Private Sub plantComboBox\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles plantComboBox.SelectedIndexChanged

quantityTextbox.Focus() 'CHANGE FOCUS TO QUANTITY TEXTBOX

End Sub

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

'ADD ITEMS TO COMBO BOX

plantComboBox.Items.Add("Flower-Iris")

plantComboBox.Items.Add("Flower-Lilac")

plantComboBox.Items.Add("Flower-Marigold")

plantComboBox.Items.Add("Herb-Basil")

plantComboBox.Items.Add("Herb-Mint")

plantComboBox.Items.Add("Shrub-Juniper")

plantComboBox.Items.Add("Shrub-Boxwood")

plantComboBox.Items.Add("Tree-Adler")

plantComboBox.Items.Add("Tree-Ash")

plantComboBox.Items.Add("Tree-Pine")

plantComboBox.Items.Add("Tree-Oak")

plantComboBox.Items.Add("Tree-Olive")

totalPurchasePrice = 0.0 'SET TOTAL PRICE TO $0.00

purchaseTotalLabel.Text = totalPurchasePrice.ToString("C", CultureInfo.GetCultureInfo("en-us"))

'FORMAT AND OUTPUT DOLLAR FORMAT

End Sub

Private Sub addItemButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles addItemButton.Click

Dim pricePerItem As Decimal 'PRICE PER ITEM

Dim plantQuantity As Integer = 0 'QUANTITY OF PLANTS ENTERED

If CInt(quantityTextbox.Text) > 0 Then 'IF NUMBER IS NOT

plantQuantity = CInt(quantityTextbox.Text) 'GET NUMBER FROM TEXTBXOX, CONVERT, PLACE IN VARIABLE

Else

plantQuantity = 0

End If

If plantComboBox.SelectedIndex <> -1 And plantQuantity > -1 Then

'IF NO COMBOBOX VALUE SELECT AND PLANT QUANTITY IS GREATER THAN -1

plantsListbox.Items.Add(plantComboBox.SelectedItem) 'ADD ITEM NAME TO LISTBOX

If plantComboBox.SelectedIndex = 0 Then 'FLOWER-IRIS

If discountCheckBox.Checked = True Then

pricePerItem = 3.84 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 4.37 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 1 Then 'FLOWER-LILAC

If discountCheckBox.Checked = True Then

pricePerItem = 4.49 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 5.22 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 2 Then 'FLOWER-MARIGOLD

If discountCheckBox.Checked = True Then

pricePerItem = 1.6 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 1.73 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 3 Then 'HERB-BASIL

If discountCheckBox.Checked = True Then

pricePerItem = 1.26 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 1.88 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 4 Then 'HERB-MINT

If discountCheckBox.Checked = True Then

pricePerItem = 2.26 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 2.88 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 5 Then 'SHRUB-JUNIPER

If discountCheckBox.Checked = True Then

pricePerItem = 3.72 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 4.18 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 6 Then 'SHRUB-BOXWOOD

If discountCheckBox.Checked = True Then

pricePerItem = 8.7 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 9.22 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 7 Then 'TREE-ADLER

If discountCheckBox.Checked = True Then

pricePerItem = 34.6 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 39.88 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 8 Then 'TREE-ASH

If discountCheckBox.Checked = True Then

pricePerItem = 38.6 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 43.8 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 9 Then 'TREE-PINE

If discountCheckBox.Checked = True Then

pricePerItem = 26.94 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 28.6 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 10 Then 'TREE-OAK

If discountCheckBox.Checked = True Then

pricePerItem = 29.32 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 33.84 'APPLY STANDARD PRICE

End If

ElseIf plantComboBox.SelectedIndex = 11 Then 'TREE-OLIVE

If discountCheckBox.Checked = True Then

pricePerItem = 17.24 'APPLY DISCOUNTED PRICE

Else

pricePerItem = 19.2 'APPLY STANDARD PRICE

End If

End If

quantityListbox.Items.Add(plantQuantity) 'ADD TO QUANTITY LIST

priceListbox.Items.Add(pricePerItem.ToString("C", CultureInfo.GetCultureInfo("en-us")))

'ADD TO PRICE LIST

totalListbox.Items.Add((pricePerItem \* plantQuantity).ToString("C", CultureInfo.GetCultureInfo("en-us")))

'ADD TO TOTAL LIST

totalPurchasePrice = totalPurchasePrice + (pricePerItem \* plantQuantity)

'TOTAL LIST BOX

purchaseTotalLabel.Text = ((totalPurchasePrice \* taxRate) + totalPurchasePrice).ToString("C", CultureInfo.GetCultureInfo("en-us")) 'UPDATE QUOTATED

End If

End Sub

Private Sub newQuotationButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles newQuotationButton.Click

quantityTextbox.Clear() 'CLEAR QUANTITY TEXTBOX

plantsListbox.Items.Clear() 'CLEAR PLANTS LISTBOX

quantityListbox.Items.Clear() 'CLEAR QUANTITY LISTBOX

priceListbox.Items.Clear() 'CLEAR PRICE LISTBOX

totalListbox.Items.Clear() 'CLEAR PRICE TOTAL LISTBOX

totalPurchasePrice = 0.0 'CHANGE TOTAL PRICE TO $0.00

plantComboBox.Focus() 'CHANGE FOCUS TO PLANT COMBO BOX

purchaseTotalLabel.Text = totalPurchasePrice.ToString("C", CultureInfo.GetCultureInfo("en-us"))

End Sub

Public Function writeToCSVFile(ByVal strData As String, ByVal FullPath As String, Optional ByVal ErrInfo As String = "") As Boolean

Dim bAns As Boolean = False

Dim fileWriter As StreamWriter

Try

fileWriter = New StreamWriter(FullPath) 'CREATE NEW (IF IT DOESN'T ALREADY EXIST) AND OPEN FILE

fileWriter.Write(strData) 'WRITE DATA TO FILE

fileWriter.Close() 'CLOSE FILE

bAns = True

Catch Ex As Exception

ErrInfo = Ex.Message 'SHOW ERROR MESSAGE

End Try

Return bAns

End Function

Private Sub writeQuotationCSVButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles writeQuotationButton.Click

Dim writeToCSVFileString As String = ""

Dim iCounter As Integer

writeToCSVFileString = "Plant, Quantity, Price Per Item, Item Total, Final Price (including Tax)" + vbNewLine

For iCounter = 0 To (plantsListbox.Items.Count - 1)

writeToCSVFileString += CStr(plantsListbox.Items.Item(iCounter)) + ", "

'WRITE ITEMS IN PLANT LIST BOX TO FILE

writeToCSVFileString += CStr(quantityListbox.Items.Item(iCounter)) + ", "

'WRITE ITEMS IN QUANTITY LIST BOX TO FILE

writeToCSVFileString += CStr(priceListbox.Items.Item(iCounter)) + ", "

'WRITE ITEMS IN PRICE PER ITEM LIST BOX TO FILE

writeToCSVFileString += CStr(totalListbox.Items.Item(iCounter)) + ", "

'WRITE ITEMS IN TOTAL LIST BOX TO FILE

writeToCSVFileString += vbNewLine

'WRITE CARRIAGE RETURN TO FILE

Next

writeToCSVFileString += ",,,," + CStr(purchaseTotalLabel.Text)

writeToCSVFile(writeToCSVFileString, "price\_quotation.txt")

End Sub

Private Sub removeItemButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles removeItemButton.Click

Dim priceVoid As Decimal 'PRICE OF ITEM VOIDED

Dim voidableItem As Integer 'INDEX OF ITEM TO VOID

If (plantsListbox.SelectedIndex <> -1) Then

voidableItem = plantsListbox.SelectedIndex

'IF ITEM IS SELECTED IN PLANTS LISTBOX, THEN USE ITS INDEX

ElseIf totalListbox.SelectedIndex <> -1 Then

voidableItem = totalListbox.SelectedIndex

'IF ITEM IS SELECTED IN TOTAL LISTBOX, THEN USE ITS INDEX

ElseIf quantityListbox.SelectedIndex <> -1 Then

voidableItem = quantityListbox.SelectedIndex

'IF ITEM IS SELECTED IN QUANTITY LISTBOX, THEN USE ITS INDEX

ElseIf priceListbox.SelectedIndex <> -1 Then

voidableItem = priceListbox.SelectedIndex

'IF ITEM IS SELECTED IN QUANTITY LISTBOX, THEN USE ITS INDEX

End If

priceVoid = CDec(totalListbox.Items.IndexOf(voidableItem)) 'GET PRICE OF ITEM TO BE VOIDED

priceListbox.Items.RemoveAt(voidableItem) 'REMOVE ITEM FROM PRICE LIST BOX

totalListbox.Items.RemoveAt(voidableItem) 'REMOVE ITEM FROM TOTAL LIST BOX

quantityListbox.Items.RemoveAt(voidableItem) 'REMOVE ITEM FROM QUANTITY LIST BOX

plantsListbox.Items.RemoveAt(voidableItem) 'REMOVE ITEM FROM PLANTS LIST BOX

totalPurchasePrice = totalPurchasePrice - priceVoid 'SUBTRACT ITEM PRICE FROM TOTAL

purchaseTotalLabel.Text = (totalPurchasePrice \* taxRate).ToString("C", CultureInfo.GetCultureInfo("en-us"))

'UPDATE QUOTATED

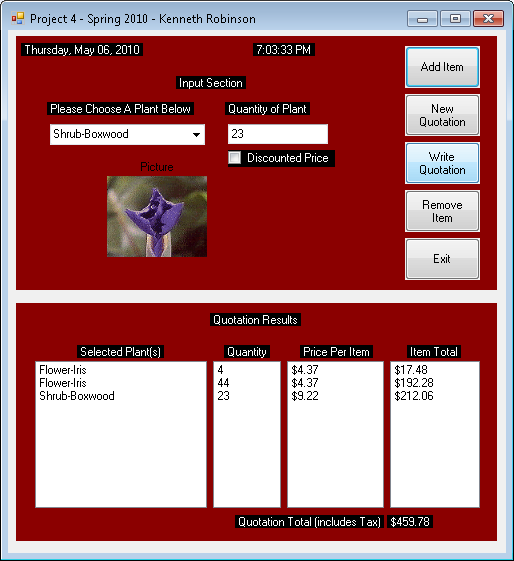
End Sub

Private Sub exitButton\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles exitButton.Click

Me.Close() 'EXIT PROGRAM

End Sub

End Class



PROJECT 5

Imports System.IO

Imports System.Globalization

Public Class project5

Private productPriceArray() As Double = New Double(10) {}

'ARRAY THAT HOLDS PRICES OF PRODUCTS

Private Sub calculateTotal()

Dim purchaseTotal As Decimal = 0.0 'INITIALIZE PURCHASE PRICE TO $0.00

Dim currentItem As Integer 'CURRENT ITEM NUMBER IN LIST BOX

For currentItem = 0 To (extPrice\_Listbox.Items.Count - 1)

'GO THROUGH EXTENDED PRICE AND TAX LIST BOXES AND ADD VALUES

purchaseTotal = purchaseTotal + CDec(extPrice\_Listbox.Items(currentItem)) + \_

CDec(tax\_ListBox.Items(currentItem))

Next

totalCost\_Label.Text = (purchaseTotal).ToString("C", CultureInfo.GetCultureInfo("en-us"))

'AFTER ADDING ALL VALUES, DISPLAY TOTAL

End Sub

Private Sub project5\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Dim inputLineFromFile As String

Dim currentItem As Integer = 0 'DECLARE AND SET CURRENT ITEM TO 0

Dim position, length, first, last As Integer

Dim inputFile As System.IO.StreamReader = System.IO.File.OpenText("office.txt")

'OPEN INPUT FILE

Try 'TRY TO OPEN FILE AND IF UNABLE, THROW EXCEPTION MESSAGE

While (inputFile.Peek <> -1) 'READ UNTIL THEN END OF FILE

inputLineFromFile = inputFile.ReadLine 'READ 1 LINE OF FILE

position = inputLineFromFile.IndexOf(",") 'GET POSITION NUMBER WHERE COMMA IS

length = inputLineFromFile.Length 'TOTAL LENGTH OF LINE

first = position + 1

last = length - first

availProducts\_ListBox.Items.Add(inputLineFromFile.Substring(0, position)) 'ADD ITEM NAME TO PRODUCTS AVAILABLE

productPriceArray(currentItem) = CDec(inputLineFromFile.Substring(first, last))

'PUT VALUE IN PRICE VARIABLE INTO ARRAY

currentItem += 1 'INCREMENT TO GET THE NEXT ITEM

End While

inputFile.Close() 'CLOSE INPUT FILE

Catch ex As Exception

Throw New Exception(ex.Message) 'THROWS EXCEPTION ERROR MESSAGE

End Try

End Sub

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

dateDisplay\_Label.Text = DateTime.Now.ToLongDateString() 'UDPATES WITH CURRENT DATE

timeDisplay\_Label.Text = DateTime.Now.ToLongTimeString() 'UDPATES WITH CURRENT TIME

End Sub

Private Sub availProducts\_ListBox\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles availProducts\_ListBox.SelectedIndexChanged

quantity\_Textbox.Focus() 'CHANGE FOCUS TO QUANTITY TEXT BOX AFTER SELECTING ITEM

End Sub

Private Sub addPurchase\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles addPurchase\_Button.Click

Dim extendedPrice, salesPrice As Decimal

If ((availProducts\_ListBox.SelectedIndex <> -1) And (CStr(quantity\_Textbox.Text)) > -1) Then

product\_ListBox.Items.Add(availProducts\_ListBox.Text) 'ADD NAME OF PRODUCT TO PRODUCT LISTBOX

quantity\_ListBox.Items.Add(quantity\_Textbox.Text) 'GETS QUANTITY FROM TEXTBOX

price\_ListBox.Items.Add((productPriceArray(availProducts\_ListBox.SelectedIndex)).ToString("C", CultureInfo.GetCultureInfo("en-us")))

'ADD QUANTITY TO QUANTITY LISTBOX

extendedPrice = (CDec(quantity\_Textbox.Text)) \* productPriceArray(availProducts\_ListBox.SelectedIndex)

'CALCULATE THE EXTENDED PRICE

extPrice\_Listbox.Items.Add((extendedPrice).ToString("C", CultureInfo.GetCultureInfo("en-us")))

'ADD EXTENDED PRICE TO EXTENDED PRICE LISTBOX

salesPrice = CDec(extendedPrice \* 0.05) 'CALCULATES SALES TAX

tax\_ListBox.Items.Add((salesPrice).ToString("C", CultureInfo.GetCultureInfo("en-us")))

'ADDS SALES TAX AS TAX LISTBOX

calculateTotal() 'CALLS FUNCTIONS TO CALCULATE THE TOTAL

quantity\_Textbox.Clear() 'CLEAR THE QUANTITY LIST BOX

availProducts\_ListBox.SelectedIndex = -1 'UNSELECT THE ITEM IN THE AVAILABLE PRODUCTS LISTBOX

availProducts\_ListBox.Focus() 'CHANGE THE FOCUS BACK TO THE AVAILABLE PRODUCTS LISTBOX

End If

End Sub

Private Sub exit\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles exit\_Button.Click

Me.Close() 'EXITS PROGRAM

End Sub

Private Sub removeItem\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles removeItem\_Button.Click

Dim indexNumber As Integer = product\_ListBox.SelectedIndex 'GET INDEX NUMBER OF ITEM SELECTED

If indexNumber > -1 Then

product\_ListBox.Items.RemoveAt(indexNumber) 'REMOVE PRODUCT IN PRODCUT IN LISTBOX

quantity\_ListBox.Items.RemoveAt(indexNumber) 'REMOVE QUANTITY IN QUANTITY LISTBOX

price\_ListBox.Items.RemoveAt(indexNumber) 'REMOVE PRICE ITEM IN PRICE LISTBOX

extPrice\_Listbox.Items.RemoveAt(indexNumber) 'REMOVE EXTENDED PRICE ITEM IN EXTENDED PRICE LISTBOX

tax\_ListBox.Items.RemoveAt(indexNumber) 'REMOVE TAX ITEM IN TAX LISTBOX

calculateTotal() 'CALCULATES TOTAL AND DISPLAYS IT

End If

End Sub

Private Sub reset\_Button\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles reset\_Button.Click

availProducts\_ListBox.SelectedIndex = -1 'RESET AVAILABLE PRODCUTS LISTBOX

product\_ListBox.Items.Clear() 'CLEAR PRODUCT LISTBOX

quantity\_ListBox.Items.Clear() 'CLEAR QUANTITY LISTBOX

price\_ListBox.Items.Clear() 'CLEAR PRICE LISTBOX

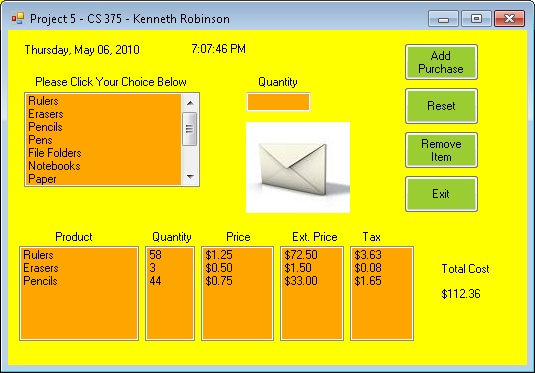
extPrice\_Listbox.Items.Clear() 'CLEAR EXTENDED PRICE LISTBOX

tax\_ListBox.Items.Clear() 'CLEAR TAX LISTBOX

quantity\_Textbox.Clear() 'CLEAR QUANTITY TEXTBOX

totalCost\_Label.Text = (0.0).ToString("c", CultureInfo.GetCultureInfo("en-us"))

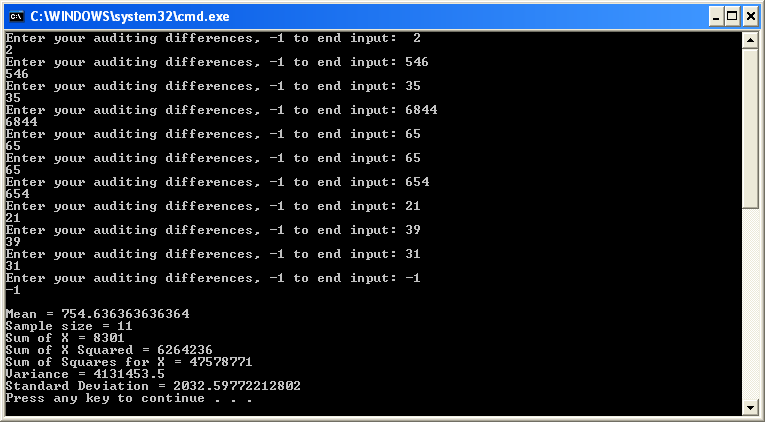
'CHANGE TOTAL COST LABEL TO $0.00

 End Sub

End Class

HOMEWORK 1

Problem 1



Module Module1

Private Sub cmdComputeStatistics\_Click()

Dim variance, standard\_deviation, X\_Bar As Double, n, sum As Integer

Statistics(sum, standard\_deviation, variance, n)

End Sub

Private Sub Statistics(ByVal sumx As Integer, ByVal sd As Double, ByVal var As Double, ByVal n As Integer)

Dim ssx, sxsq As Long, x As Integer, mean As Double

While x <> (-1)

If x <> -1 Then

Console.Write("Enter your auditing differences, -1 to end input: ")

x = CDbl(Console.ReadLine)

ssx = ssx + x ^ 2

sumx = sumx + x

n = n + 1

Console.WriteLine(x)

End If

End While

mean = sumx / n

sxsq = (sumx ^ 2) / n

var = (ssx - sxsq) / (n - 1)

sd = Math.Sqrt(var)

Console.WriteLine()

Console.WriteLine("Mean = {0}", mean)

Console.WriteLine("Sample size = {0}", n)

Console.WriteLine("Sum of X = {0}", sumx)

Console.WriteLine("Sum of X Squared = {0}", sxsq)

Console.WriteLine("Sum of Squares for X = {0}", ssx)

Console.WriteLine("Variance = {0}", var)

Console.WriteLine("Standard Deviation = {0}", sd)

End Sub

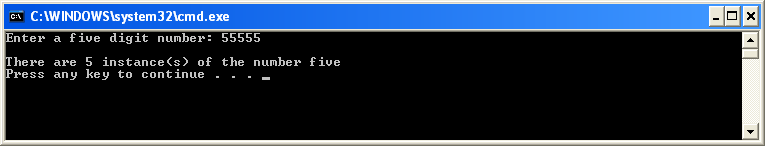
Sub Main()

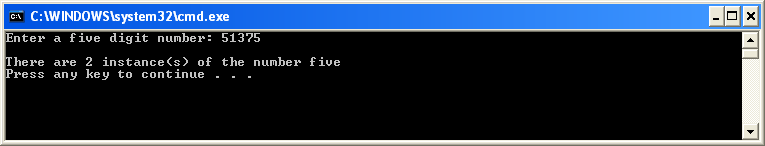
cmdComputeStatistics\_Click()

End Sub

End Module

Problem 2

****

****

Module Module1

Dim user\_input As String, num\_of\_input, number\_found, start\_position, position As Integer

Sub Main()

Console.Write("Enter a five digit number: ")

user\_input = CStr(Console.ReadLine)

start\_position = 1

number\_found = 0

While start\_position < 6

position = InStr(start\_position, user\_input, "5") 'RETURNS POSITION OF 5 IF FOUND

If position > 0 Then

number\_found = number\_found + 1

End If

If position > start\_position Then

'IF THE START POSITION IS GREATER THAN THE CURRENT POSITION, GO TO THE NEXT POSITION AFTER THE 5

start\_position = position + 1

Else

'INCREMENT START POSITION

start\_position = start\_position + 1

End If

End While

Console.WriteLine("")

Console.WriteLine("There are {0} instance(s) of the number five", number\_found)

End Sub

End Module

HOMEWORK 2

Imports System.IO

Public Module GlobalMembersMain

Public Sub Load2DArray(ByVal ls As Integer(,), ByVal r As Integer, ByRef input1 As StreamReader)

For i As Integer = 0 To (r - 1)

For j As Integer = 0 To 3

'input1 = ls(i, j)

Next

Next

Exit Sub

End Sub

Public Sub Row\_Totals(ByVal exp As Integer(,), ByVal rt As Integer(), ByVal c As Integer)

Dim sum As Integer = 0

For i As Integer = 0 To (c - 1)

For j As Integer = 0 To (c - 1)

sum = sum + exp(i, j)

Next

rt(i) = sum

sum = 0

Next

Exit Sub

End Sub

Public Sub Column\_Totals(ByVal e As Integer(,), ByVal ct As Integer(), ByVal r As Integer)

Dim num As Integer = 0

Dim total As Integer = 0

For m As Integer = 0 To (r - 1)

For n As Integer = 0 To (r - 1)

ct(n) = ct(n) + e(m, n)

Next

Next

Exit Sub

End Sub

Public Sub Treatment\_Totals(ByVal e As Integer(,), ByVal t As Integer(), ByVal r As Integer)

t(0) = e(0, 0) + e(1, 1) + e(3, 2) + e(2, 3)

t(1) = e(1, 0) + e(3, 1) + e(2, 2) + e(0, 3)

t(2) = e(2, 0) + e(0, 1) + e(1, 2) + e(3, 3)

t(3) = e(3, 0) + e(2, 1) + e(0, 2) + e(1, 3)

Exit Sub

End Sub

Public Sub Mean\_Totals(ByVal rt As Integer(), ByVal c As Integer(), ByVal t As Integer(), ByVal rm As Double(), ByVal cm As Double(), ByVal tm As Double(), \_

ByVal r As Integer, ByRef gm As Double)

Dim sum As Double = 0

For i As Integer = 0 To (r - 1)

rm(i) = CDbl((rt(i))) / r

cm(i) = CDbl((c(i))) / r

tm(i) = CDbl((t(i))) / r

sum = sum + rm(i)

Next

gm = sum / 4.0R

Exit Sub

End Sub

Public Sub Statistical\_Analysis(ByVal e As Integer(,), ByVal rm As Double(), ByVal cm As Double(), ByVal tm As Double(), ByVal r As Integer, ByVal gm As Double)

Dim ssr As Double = 0.0R

Dim ssc As Double = 0.0R

Dim sst As Double = 0.0R

Dim tsum As Double = 0.0R

Dim tss As Double = 0.0R

Dim tsq As Double = 0.0R

Dim sse As Double

Dim msr As Double

Dim msc As Double

Dim mst As Double

Dim mse As Double

Dim f As Double

Dim dfr As Integer

Dim dfc As Integer

Dim dft As Integer

Dim dfe As Integer

Dim dfst As Integer

For i As Integer = 0 To (r - 1)

ssr = ssr + (Math.Pow((rm(i) - gm), 2)) \* r

ssc = ssc + (Math.Pow((cm(i) - gm), 2)) \* r

sst = sst + (Math.Pow((tm(i) - gm), 2)) \* r

Next

For m As Integer = 0 To (r - 1)

For n As Integer = 0 To (r - 1)

'tsq=tsq+pow(e[m][n],2);

tsum = tsum + e(m, n)

Next

Next

tsum = Math.Pow(tsum, 2) / 16.0R

tss = tsq - tsum

dfr = r - 1

dfc = r - 1

dft = r - 1

dfst = (r \* r) - 1

sse = tss - (ssr + ssc + sst)

dfe = dfst - (dfr + dfc + dft)

msr = ssr / dfr

msc = ssc / dfc

mst = sst / dft

mse = sse / dfe

f = mst / mse

Console.Write("Source")

Console.Write(ControlChars.Tab)

Console.Write(ControlChars.Tab)

Console.Write("SS")

Console.Write(ControlChars.Tab)

Console.Write("df")

Console.Write(ControlChars.Tab)

Console.Write("MS")

Console.Write(ControlChars.Tab)

Console.Write(ControlChars.Tab)

Console.Write("F")

Console.Write(vbLf)

Console.Write("Rows = ")

Console.Write(ssr)

Console.Write(ControlChars.Tab)

Console.Write(dfr)

Console.Write(ControlChars.Tab)

Console.Write(msr)

Console.Write(ControlChars.Tab)

Console.Write(vbLf)

Console.Write("Columns = ")

Console.Write(ssc)

Console.Write(ControlChars.Tab)

Console.Write(dfc)

Console.Write(ControlChars.Tab)

Console.Write(msc)

Console.Write(ControlChars.Tab)

Console.Write(vbLf)

Console.Write("Treatment = ")

Console.Write(sst)

Console.Write(ControlChars.Tab)

Console.Write(dft)

Console.Write(ControlChars.Tab)

Console.Write(mst)

Console.Write(ControlChars.Tab)

Console.Write(ControlChars.Tab)

Console.Write(f)

Console.Write(vbLf)

Console.Write("Error = ")

Console.Write(sse)

Console.Write(ControlChars.Tab)

Console.Write(dfe)

Console.Write(ControlChars.Tab)

Console.Write(mse)

Console.Write(ControlChars.Tab)

Console.Write(vbLf)

Console.Write("Total = ")

Console.Write(tss)

Console.Write(ControlChars.Tab)

Console.Write(dfst)

Console.Write(ControlChars.Tab)

Console.Write(ControlChars.Tab)

Console.Write(vbLf)

Exit Sub

End Sub

Sub Main()

Const row As Integer = 4

Const column As Integer = 4

Dim experiment As Integer(,) = New Integer(3, 3) {}

Dim i As Integer

Dim j As Integer

Dim row\_total As Integer() = New Integer(3) {}

Dim column\_total As Integer() = {0, 0, 0}

Dim treatment\_total As Integer() = New Integer(3) {}

Dim row\_means As Double() = New Double(3) {}

Dim column\_means As Double() = New Double(3) {}

Dim treatment\_means As Double() = New Double(3) {}

Dim grand\_mean As Double = 0.0R

If File.Exists("experiment2.txt") Then

Dim input1 As StreamReader = File.OpenText("experiment2.txt")

Load2DArray(experiment, column, input1)

Console.Write("The data for the Latin Square Design are :")

Console.Write(vbLf)

For i = 0 To (row - 1)

For j = 0 To (column - 1)

Console.Write(experiment(i, j))

Console.Write(" ")

Next

Console.Write(vbLf)

Next

Row\_Totals(experiment, row\_total, column)

Column\_Totals(experiment, column\_total, row)

Treatment\_Totals(experiment, treatment\_total, row)

Console.Write("Row totals: ")

Console.Write(" ")

Console.Write("Column totals: ")

Console.Write(" ")

Console.Write("Treatment totals")

Console.Write(vbLf)

For i = 0 To (row - 1)

Console.Write(ControlChars.Tab)

Console.Write(row\_total(i))

Console.Write(ControlChars.Tab)

Console.Write(column\_total(i))

Console.Write(ControlChars.Tab)

Console.Write(ControlChars.Tab)

Console.Write(treatment\_total(i))

Console.Write(vbLf)

Next

Mean\_Totals(row\_total, column\_total, treatment\_total, row\_means, column\_means, treatment\_means, \_

row, grand\_mean)

Console.Write("Row means: ")

Console.Write(" ")

Console.Write("Column means: ")

Console.Write(" ")

Console.Write("Treatment means")

Console.Write(vbLf)

For i = 0 To (row - 1)

Console.Write(ControlChars.Tab)

Console.Write(row\_means(i))

Console.Write(ControlChars.Tab)

Console.Write(column\_means(i))

Console.Write(ControlChars.Tab)

Console.Write(ControlChars.Tab)

Console.Write(treatment\_means(i))

Console.Write(vbLf)

Next

Console.Write("Grand Mean = ")

Console.Write(grand\_mean)

Console.Write(vbLf)

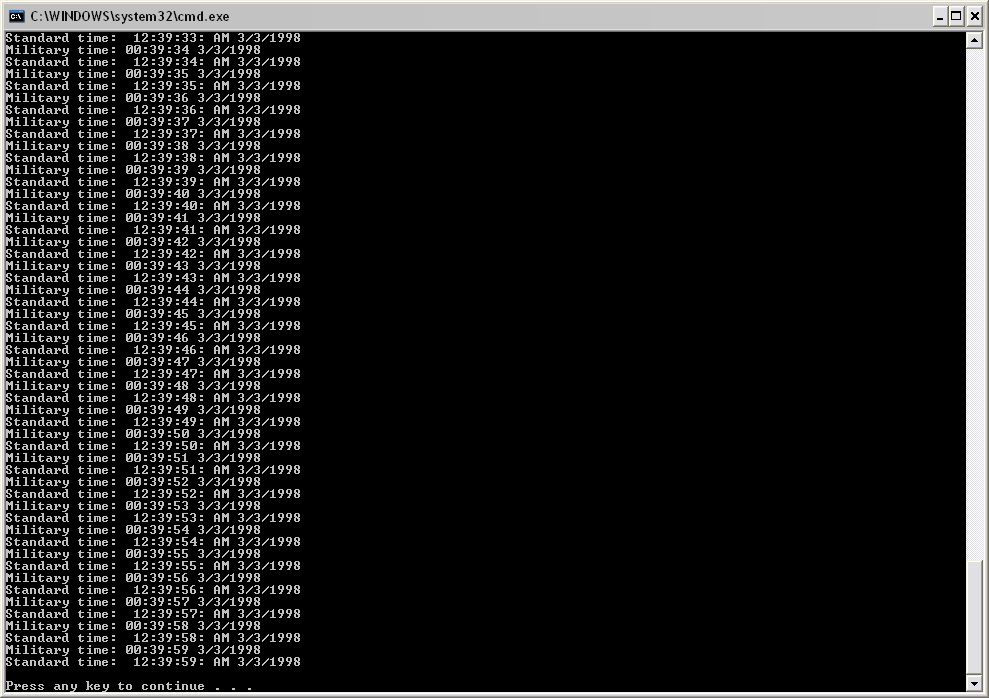
Statistical\_Analysis(experiment, row\_means, column\_means, treatment\_means, row, grand\_mean)

End If

End Sub

End Module

HOMEWORK 3



Public Class GlobalMembersHw

Shared Function Main() As Integer

Const MAXTICKS As Integer = 3000

Dim d As New DateTime(3, 2, 1998, 23, 50, 0)

For ticks As Integer = 1 To MAXTICKS

Console.Write("Military time: ")

d.printMilitary()

Console.Write("Standard time: ")

d.printStandard()

d.tick()

Next ticks

Console.Write(ControlChars.Lf)

Return 0

End Function

End Class

Public Class DateTime

Public Sub New(Optional ByVal m As Integer = 1, Optional ByVal d As Integer = 1, \_

Optional ByVal y As Integer = 1900, Optional ByVal hr As Integer = 0, \_

Optional ByVal min As Integer = 0, Optional ByVal sec As Integer = 0)

setDate(m, d, y)

setTime(hr, min, sec)

End Sub

Public Sub setDate(ByVal mo As Integer, ByVal dy As Integer, ByVal yr As Integer)

setMonth(mo)

setDay(dy)

setYear(yr)

End Sub

Public Sub setMonth(ByVal m As Integer)

If m <= 12 AndAlso m >= 1 Then

month = m

Else

month = 1

End If

End Sub

Public Sub setDay(ByVal d As Integer)

If month = 2 AndAlso leapYear() Then

If d <= 29 AndAlso d >= 1 Then

day = d

Else

day = 1

End If

Else

If d <= monthDays() AndAlso d >= 1 Then

day = d

Else

day = 1

End If

End If

End Sub

Public Sub setYear(ByVal y As Integer)

If y <= 2050 AndAlso y >= 1900 Then

year = y

Else

year = 1900

End If

End Sub

Public Function getMonth() As Integer

Return month

End Function

Public Function getDay() As Integer

Return day

End Function

Public Function getYear() As Integer

Return year

End Function

Public Sub nextDay()

day += 1

setDay(day)

If day = 1 Then

month += 1

setMonth(month)

If month = 1 Then

year += 1

setYear(year)

End If

End If

End Sub

Public Sub setTime(ByVal hr As Integer, ByVal min As Integer, ByVal sec As Integer)

setHour(hr)

setMinute(min)

setSecond(sec)

End Sub

Public Sub setHour(ByVal h As Integer)

If h >= 0 AndAlso h < 24 Then

hour = h

Else

hour = 0

End If

End Sub

Public Sub setMinute(ByVal m As Integer)

If m >= 0 AndAlso m < 60 Then

minute = m

Else

minute = 0

End If

End Sub

Public Sub setSecond(ByVal s As Integer)

If s >= 0 AndAlso s < 60 Then

second = s

Else

second = 0

End If

End Sub

Public Function getHour() As Integer

Return hour

End Function

Public Function getMinute() As Integer

Return minute

End Function

Public Function getSecond() As Integer

Return second

End Function

Public Sub printMilitary()

If hour < 10 Then

Console.Write("0")

Console.Write(hour)

Console.Write(":"c)

Else

Console.Write("")

Console.Write(hour)

Console.Write(":"c)

End If

If minute < 10 Then

Console.Write("0")

Console.Write(minute)

Console.Write(":"c)

Else

Console.Write("")

Console.Write(minute)

Console.Write(":"c)

End If

If second < 10 Then

Console.Write("0")

Console.Write(second)

Console.Write(" ")

Else

Console.Write("")

Console.Write(second)

Console.Write(" ")

End If

Console.Write(month)

Console.Write("/"c)

Console.Write(day)

Console.Write("/"c)

Console.Write(year)

Console.Write(ControlChars.Lf)

End Sub

Public Sub printStandard()

If hour Mod 12 = 0 Then

Console.Write(12)

Console.Write(":"c)

Else

Console.Write(hour Mod 12)

Console.Write(":"c)

End If

If minute < 10 Then

Console.Write("0")

Console.Write(minute)

Console.Write(":"c)

Else

Console.Write("")

Console.Write(minute)

Console.Write(":"c)

End If

If second < 10 Then

Console.Write("0")

Console.Write(second)

Console.Write(":"c)

Else

Console.Write("")

Console.Write(second)

Console.Write(":"c)

End If

If hour < 12 Then

Console.Write(" AM ")

Else

Console.Write(" PM ")

End If

Console.Write(month)

Console.Write("/"c)

Console.Write(day)

Console.Write("/"c)

Console.Write(year)

Console.Write(ControlChars.Lf)

End Sub

Public Function monthDays() As Integer

Dim days() As Integer = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

If month = 2 AndAlso leapYear() Then

Return 29

Else

Return days((month - 1))

End If

End Function

Public Sub tick()

second += 1

setSecond(second)

If second = 0 Then

minute += 1

setMinute(minute)

If minute = 0 Then

hour += 1

setHour(hour)

If hour = 0 Then

nextDay()

End If

End If

End If

End Sub

Public Function leapYear() As Boolean

If year Mod 400 = 0 OrElse (year Mod 4 = 0 AndAlso year Mod 100 <> 0) Then

Return True

Else

Return False

End If

End Function

Private month As Integer

Private day As Integer

Private year As Integer

Private hour As Integer

Private minute As Integer

Private second As Integer

End Class

HOMEWORK 4

PROBLEM 1

Module HW4Sp10

Sub Main()

Dim sentence As String

Console.Write(" Enter a sentence: ")

'Mom and Dad are coming at noon there names are Bob and Amanda

'sentence = LCase(Console.ReadLine)

sentence = (Console.ReadLine).ToLower()

Call Tokenize(sentence)

End Sub

Private Sub Tokenize(ByVal sentence As String)

Dim s() As String, j As Integer

's = Split(sentence)

s = sentence.Split(" ")

For j = 0 To s.GetUpperBound(0)

If Unknown(s(j)) Then

Console.WriteLine(s(j))

End If

Next j

End Sub

Private Function Unknown(ByVal s As String) As Boolean

Dim length As Integer, j As Integer, first As String, last As String

'length = Len(s)

length = s.Length

If length = 1 Then

Return False

Exit Function

End If

For j = 1 To length

Dim startCheck As Integer = Mid(s, j, 1)

Dim endCheck As Integer = Mid(s, length, 1)

If s.Substring(1) <> s.Substring(1) Then

If startCheck <> endCheck Then

mid()

Return False

Exit Function

End If

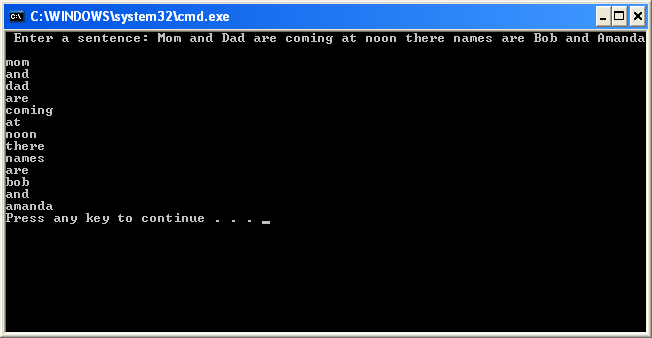
length = length - 1

Next j

Return True

End Function

End Module



PROBLEM 2

Module Module1

Sub Main()

Dim menuOption As String = "0"

Dim decryptedText As String

Dim encryptedText As String

Dim convertCharacter As Char

Dim convertNumber As Integer

Dim stringPosition As Integer '

Console.WriteLine("Text Encryption Program based on ROT13 Encryption Method.")

Do Until menuOption = "3" 'KEEP RUNNING PROGRAM UNTIL TOLD TO EXIT

Console.WriteLine("-----MAIN MENU-----")

Console.WriteLine(" 1) Encrypt text") 'DISPLAY MENU CHOICES

Console.WriteLine(" 2) Decrypt text")

Console.WriteLine(" 3) Exit")

Console.Write("Choose an option from above: ")

menuOption = Console.ReadLine

If menuOption = "1" Then 'ENCRYPTER

encryptedText = "" 'CLEAR STRING IN CASE OF PREVIOUS USAGE

Console.WriteLine("") 'BLANK LINE

Console.Write("Enter the text that you want to encrypt: ")

decryptedText = Console.ReadLine 'TEXT IN RAW FORM

Console.WriteLine("Encrypting...")

Console.WriteLine("") 'BLANK LINE

For stringPosition = 0 To (decryptedText.Length - 1)

'GET CHARACTER FROM STRING AT CURRENT POSITION

convertCharacter = decryptedText.Substring(stringPosition, 1)

convertNumber = AscW(convertCharacter) 'CONVERT CHARACTER TO NUMBER

'CONVERT NUBMERS FOR ENCRPYTING

If convertNumber >= 97 And convertNumber <= 109 Then

convertNumber = convertNumber + 13

ElseIf convertNumber >= 110 And convertNumber <= 122 Then

convertNumber = convertNumber - 13

ElseIf convertNumber >= 65 And convertNumber <= 77 Then

convertNumber = convertNumber + 13

ElseIf convertNumber >= 78 And convertNumber <= 90 Then

convertNumber = convertNumber - 13

End If

'ADD ENCRYPTED CHARACTER TO ENCRYPTED STRING AND CONVERT TO LETTER

encryptedText = encryptedText + ChrW(convertNumber)

Next stringPosition 'END OF FOR LOOP

Console.WriteLine(encryptedText) 'DISPLAY ENCRYPTED TEXT

'''''END OF ENCRYPTION SECTION'''''

ElseIf menuOption = "2" Then 'DECRYPTER

decryptedText = "" 'CLEAR STRING IN CASE OF PREVIOUS USAGE

Console.WriteLine("") 'BLANK LINE

Console.Write("Enter the text that you want to decrypt: ")

encryptedText = Console.ReadLine

Console.WriteLine("Decrypting...")

Console.WriteLine("") 'BLANK LINE

For stringPosition = 0 To (encryptedText.Length - 1)

'GET CHARACTER FROM STRING AT CURRENT POSITION

convertCharacter = encryptedText.Substring(stringPosition, 1)

convertNumber = AscW(convertCharacter) 'CONVERT CHARACTER TO NUMBER

'CONVERT NUMBERS FOR ENCRYPTING

If convertNumber >= 97 And convertNumber <= 109 Then

convertNumber = convertNumber + 13

ElseIf convertNumber >= 110 And convertNumber <= 122 Then

convertNumber = convertNumber - 13

ElseIf convertNumber >= 65 And convertNumber <= 77 Then

convertNumber = convertNumber + 13

ElseIf convertNumber >= 78 And convertNumber <= 90 Then

convertNumber = convertNumber - 13

End If

'ADD DECRYPTED CHARACTER TO DECRYPTED STRING AND CONVERT TO LETTER

decryptedText = decryptedText + ChrW(convertNumber)

Next stringPosition 'END OF FOR LOOP

Console.WriteLine(decryptedText) 'DISPLAY DECRPYTED TEXT

'''''END OF DECRYPTION SECTION'''''

ElseIf menuOption = "3" Then 'EXIT PROGRAM

Exit Sub

Else

Console.WriteLine("") 'BLANK LINE

Console.WriteLine("You have picked an invalid option. Please select again.") 'ERROR MESSAGE

End If

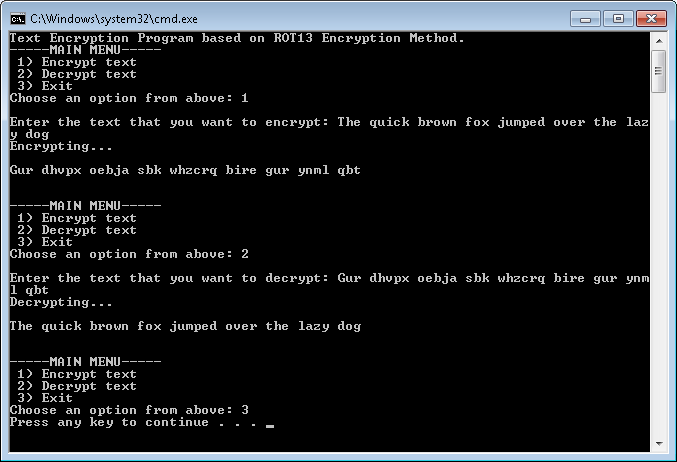
Console.WriteLine("") 'BLANK LINE

Console.WriteLine("") 'BLANK LINE

Loop

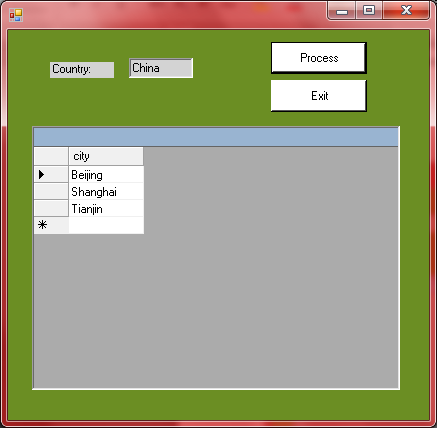
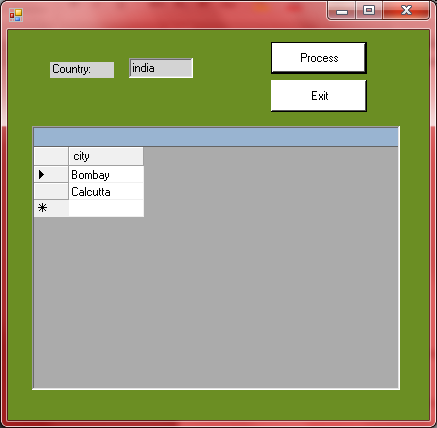
End Sub

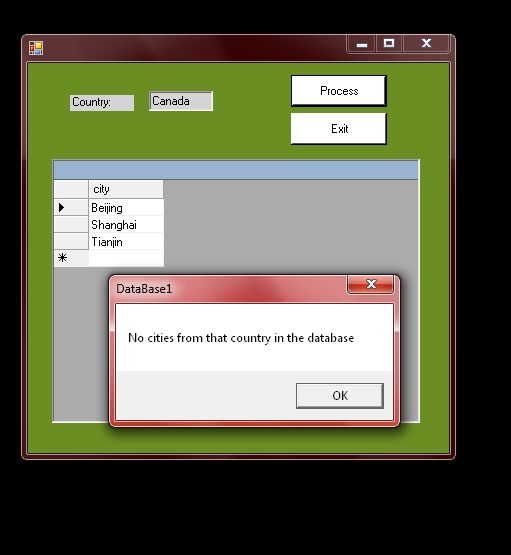
End Module



HOMEWORK 5

Problem 1





Public Class frmDatabase1

Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

Public Sub New()

MyBase.New()

InitializeComponent() 'This call is required by the Windows Form Designer.

'Add any initialization after the InitializeComponent() call

End Sub

'Form overrides dispose to clean up the component list.

Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)

If disposing Then

If Not (components Is Nothing) Then

components.Dispose()

End If

End If

MyBase.Dispose(disposing)

End Sub

'Required by the Windows Form Designer

Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

Friend WithEvents dgDisplay As System.Windows.Forms.DataGrid

Friend WithEvents btnOrderbyPop As System.Windows.Forms.Button

Friend WithEvents btnShowCurrency As System.Windows.Forms.Button

Friend WithEvents btnFindCities As System.Windows.Forms.Button

Friend WithEvents txtCountry As System.Windows.Forms.TextBox

Friend WithEvents lblCountry As System.Windows.Forms.Label

Friend WithEvents dgDisplay2 As System.Windows.Forms.DataGrid

<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

Me.dgDisplay = New System.Windows.Forms.DataGrid

Me.btnOrderbyPop = New System.Windows.Forms.Button

Me.btnShowCurrency = New System.Windows.Forms.Button

Me.btnFindCities = New System.Windows.Forms.Button

Me.txtCountry = New System.Windows.Forms.TextBox

Me.lblCountry = New System.Windows.Forms.Label

Me.dgDisplay2 = New System.Windows.Forms.DataGrid

CType(Me.dgDisplay, System.ComponentModel.ISupportInitialize).BeginInit()

CType(Me.dgDisplay2, System.ComponentModel.ISupportInitialize).BeginInit()

Me.SuspendLayout()

'

'dgDisplay

'

Me.dgDisplay.BackColor = System.Drawing.Color.White

Me.dgDisplay.DataMember = ""

Me.dgDisplay.HeaderForeColor = System.Drawing.SystemColors.ControlText

Me.dgDisplay.Location = New System.Drawing.Point(24, 96)

Me.dgDisplay.Name = "dgDisplay"

Me.dgDisplay.Size = New System.Drawing.Size(368, 264)

Me.dgDisplay.TabIndex = 0

'

'btnOrderbyPop

'

Me.btnOrderbyPop.BackColor = System.Drawing.Color.Yellow

Me.btnOrderbyPop.Location = New System.Drawing.Point(40, 56)

Me.btnOrderbyPop.Name = "btnOrderbyPop"

Me.btnOrderbyPop.Size = New System.Drawing.Size(160, 32)

Me.btnOrderbyPop.TabIndex = 1

Me.btnOrderbyPop.Text = "Order By 1995 Population"

'

'btnShowCurrency

'

Me.btnShowCurrency.BackColor = System.Drawing.Color.Yellow

Me.btnShowCurrency.Location = New System.Drawing.Point(248, 56)

Me.btnShowCurrency.Name = "btnShowCurrency"

Me.btnShowCurrency.Size = New System.Drawing.Size(96, 32)

Me.btnShowCurrency.TabIndex = 2

Me.btnShowCurrency.Text = "Show Currency"

'

'btnFindCities

'

Me.btnFindCities.BackColor = System.Drawing.Color.Yellow

Me.btnFindCities.Location = New System.Drawing.Point(528, 56)

Me.btnFindCities.Name = "btnFindCities"

Me.btnFindCities.Size = New System.Drawing.Size(96, 32)

Me.btnFindCities.TabIndex = 3

Me.btnFindCities.Text = "Find City"

'

'txtCountry

'

Me.txtCountry.BackColor = System.Drawing.Color.Yellow

Me.txtCountry.Location = New System.Drawing.Point(624, 24)

Me.txtCountry.Name = "txtCountry"

Me.txtCountry.Size = New System.Drawing.Size(64, 20)

Me.txtCountry.TabIndex = 4

Me.txtCountry.Text = ""

'

'lblCountry

'

Me.lblCountry.BackColor = System.Drawing.Color.Yellow

Me.lblCountry.Location = New System.Drawing.Point(528, 24)

Me.lblCountry.Name = "lblCountry"

Me.lblCountry.Size = New System.Drawing.Size(64, 16)

Me.lblCountry.TabIndex = 5

Me.lblCountry.Text = "Country:"

'

'dgDisplay2

'

Me.dgDisplay2.BackColor = System.Drawing.Color.White

Me.dgDisplay2.DataMember = ""

Me.dgDisplay2.HeaderForeColor = System.Drawing.SystemColors.ControlText

Me.dgDisplay2.Location = New System.Drawing.Point(480, 96)

Me.dgDisplay2.Name = "dgDisplay2"

Me.dgDisplay2.Size = New System.Drawing.Size(200, 112)

Me.dgDisplay2.TabIndex = 6

'

'frmDatabase1

'

Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)

Me.BackColor = System.Drawing.Color.Blue

Me.ClientSize = New System.Drawing.Size(720, 390)

Me.Controls.Add(Me.dgDisplay2)

Me.Controls.Add(Me.lblCountry)

Me.Controls.Add(Me.txtCountry)

Me.Controls.Add(Me.btnFindCities)

Me.Controls.Add(Me.btnShowCurrency)

Me.Controls.Add(Me.btnOrderbyPop)

Me.Controls.Add(Me.dgDisplay)

Me.Name = "frmDatabase1"

CType(Me.dgDisplay, System.ComponentModel.ISupportInitialize).EndInit()

CType(Me.dgDisplay2, System.ComponentModel.ISupportInitialize).EndInit()

Me.ResumeLayout(False)

End Sub

#End Region

Dim dt As New DataTable

Private Sub frmDatabase1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

UpdateDataGrid("Select \* From Cities")

End Sub

Private Sub btnOrderbyPop\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnOrderbyPop.Click

UpdateDataGrid("Select \* From Cities order by pop1995 ASC")

End Sub

Private Sub UpdateDataGrid(ByVal sqlStr As String)

Dim dt As New DataTable

Dim connStr As String = "Provider=Microsoft.Jet.OLEDB.4.0;" & \_

"Data Source = MEGACITIES.MDB "

Dim dataAdapter As New OleDb.OleDbDataAdapter(sqlStr, connStr)

dataAdapter.Fill(dt)

dataAdapter.Dispose()

dgDisplay.DataSource = dt

End Sub

Private Sub btnShowCurrency\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnShowCurrency.Click

UpdateDataGrid("SELECT city, Cities.country, " & \_

"Cities.pop1995, currency " & \_

"FROM Cities INNER JOIN Countries " & \_

"ON Cities.country=Countries.country " & \_

"ORDER BY city ASC")

End Sub

Private Sub btnFindCities\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnFindCities.Click

UpdateDataGrid2("SELECT city FROM Cities WHERE country = '" & txtCountry.Text & "' ORDER BY city ASC")

End Sub

Private Sub UpdateDataGrid2(ByVal sqlStr As String)

Dim dt2 As New DataTable

Dim connStr As String = "Provider=Microsoft.Jet.OLEDB.4.0;" & \_

"Data Source = MEGACITIES.MDB "

Dim dataAdapter2 As New OleDb.OleDbDataAdapter(sqlStr, connStr)

dataAdapter2.Fill(dt2)

dataAdapter2.Dispose()

dgDisplay2.DataSource = dt2

If dt2.Rows.Count = 0 Then

MsgBox("No cities from that country in the database")

Else

dgDisplay2.DataSource = dt2

End If

End Sub

End Class

Problem 2

**DataTable** – an object that contains columns and rows and is used to display organize data in a grid format

**DataAdapter** – is used to fill a dataset using SQL Commands and a database connection

**DataSource** – is a property in the DataGridView class that is used to get or set the source of the data that is being displayed.

**DataGrid** – displays items from a data source in a table format.

**CommandBuilder** – generates commands that make changes to data within a DataSet.