CENTRE FOR DIPLOMA STUDIES

MULTIMEDIA CONCEPT

LABORATORY INSTRUCTION SHEET

Course Code	**DAT 21103**
Lab Practical Title	**DATA AND OPERATORS**
Lab Practical	**6**

Guided Task 6.1: Create a Crack Password

Contextual Situation 6.1:

Create a VB program that demonstrates how to crack passwords. It can generate possible passwords and compare each of them with the actual password; and if the generated password found to be equal to the actual password, login will be successful. In the program provides as following:

1. A timer will be inserted into the form and it is used to do a repetitive job of generating the passwords. A password generating procedure `generate()` will be created and it is called by the e `Timer1_Tick()` event so that the procedure is repeated after every interval. The interval of the timer can be set in its properties window. A value of 1 is 1 millisecond and a value of 1000 is 1 second. Enabled property is set to false so that the program will only start generating the passwords after the user clicks on the Generate button.
2. `Rnd( )` is a Visual Basic 2015 Math function that generates a random number between 0 and 1. Multiplying `Rnd( )` by 100 will produce a number between 0 and 100.
3. `Int( )` is a Visual Basic 2015 function that returns an integer by ignoring the decimal part of that number. Therefore, `Int(Rnd*100)` will produce a number between 0 and 99, and the value of `Int(Rnd*100)+100` will produce a number between 100 and 199.
4. Finally, the program uses If…Then…Else to check whether the generated password is equal the actual password or not. If they are equal, the passwords generating process will be terminated by setting the `Timer1.Enabled` property to false.

Make a solution for above situation as following steps:

A. Create the user interface

Create a new project and create a form. The form should contains a Form, a Timer, 2 labels and a button.

B. Set the Properties

Set the properties of a Form, 2 labels, a Button and A Timer using prefix name. For the Timer, set the Timer's interval at 100 which is equivalent to 0.1 second.

C. Writing a program code

1. Declare the module variables as follows:

```
Dim password As Integer
Dim crackpass As Integer
```

2. Event procedure for `frmCracker_Load( )`

```
Private Sub frmCracker_Load(sender As Object, e As EventArgs)
     Handles_
MyBase.Load
```

```
password = 123
End Sub
```

3. Event procedure for `btnGenerate_click( )`

```
Private Sub btnGenerate_Click(sender As Object, e As
      EventArgs)Handles _ btnGenerate.Click
Timer1.Enabled = True
End Sub
```

4. Write a Sub Procedure `generate( )`

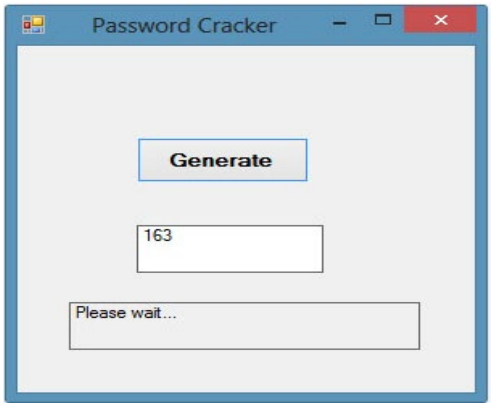
```
Sub generate( )
       crackpass = Int(Rnd( ) * 100) + 100
End Sub
```

5. Event procedure for `Timer1_Tick( )`

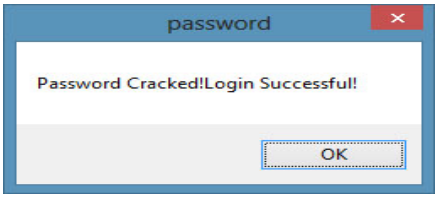
```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
      Timer1.Tick
      Call generate( )
      If crackpass = password Then
            Timer1.Enabled = False
            lblResult.Text = crackpass
            MsgBox("Password Cracked!Login Successful!")
      Else
            lblResult.Text = crackpass
            lblProcess.Text = "Please wait…"
      End If
End Sub
```

6. Run the Program

After execute the program, a button generate will be click. Then, a password is generated and will appeared at `lblResult.Text` as shown below:



If the result of generated password match with the actual password, a message box will be appear as shown below.



Guided Task 6.2: TextBox Sub Program

Contextual Situation 6.2:

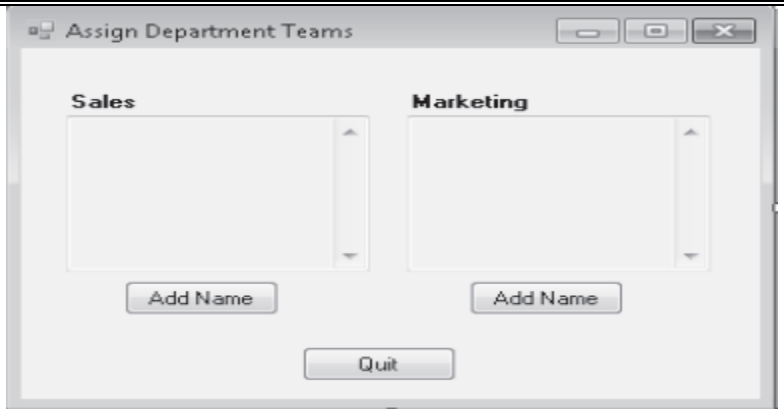
Sub procedures are often used to handle input in a program when information comes from two or more sources and needs to be in the same format. In the following exercise, you'll create a Sub procedure named `AddName` that prompts the user for input and formats the text so that it can be displayed on multiple lines in a text box. Sub procedure uses the `InputBox` function to prompt the user for an employee name. The procedure will save you programming time because you'll use it in two event procedures, each associated with a different text box. Because the procedure will be declared in a module, you'll need to type it in only one place. If you add additional forms to the project, the procedure will be available to them as well.

Make a solution for above situation as following steps:

A. Create the user interfaces

1. Create a new Windows Forms Application project named `MyTxtboxSub`. The new project is created, and a blank form opens in the Designer.
2. Use the TextBox control to create two text boxes, side by side, in the middle of the form. Today you'll make some personnel decisions, and you'll use these text boxes to hold the names of employees you'll be assigning to two departments.
3. Use the Label control to create two labels above the text boxes.These labels will hold the names of the departments.
4. Use the Button control to create three buttons:
   i. One under each text box and one at the bottom of the form.
   ii. You'll use the first two buttons to assign employees to their departments and
   iii. Last button to quit the program.

   Your form looks similar to this:



A. Set the properties

Set the properties shown in the following table for the objects on the form. Because the text boxes will contain more than one line, you'll set their Multiline Properties to True and their ScrollBars properties to Vertical. These settings are typically used when multiple lines are displayed in text boxes. You'll also set their TabStop properties to False and their ReadOnly properties to True so that the information can'to be modified.

Object	Property	Setting
Form1	Text	"Assign Department Terms"
TextBox1	Multiline	True
	Name	txtSales
	ReadOnly	True
	ScrollBars	Vertical
	TabStop	False
TextBox2	Multiline	True
	Name	txtMkt
	ReadOnly	True
	ScrollBars	Vertical
	TabStop	False
Label	Font	bold
	Name	lblSales
	Text	"Sales"
Labe2	Font	bold
	Name	lblMkt
	Text	"Marketing"
Button1	Name	btnSales
	Text	"Add Name"
Button2	Name	btnMkt
	Text	"Add Name"
Button3	Name	btnQuit

	Text	"Quit"

C. Write the program code

1. Create a **sub procedure**
   i. On the Project menu, click the Add Module… command, select the Module template, and then click Add. A new module appears in the Code Editor.
   ii. Type the following `AddName` procedure between the `Module Module 1` and `End Module` statements:

```
Sub AddName(ByVal Team As String, ByRef ReturnString As
    String)
     Dim Prompt, Nm, WrapCharacter As String
     Prompt = "Enter a " & Team & " employee."
     Nm = InputBox(Prompt, "Input Box")
     WrapCharacter = Chr(13) + Chr(10)
     ReturnString = Nm & WrapCharacter
End Sub
```

This general-purpose Sub procedure uses the `InputBox` function to prompt the user for an employee name. It receives two arguments during the procedure call: `Team`, a string containing the department name; and `ReturnString`, an empty string variable that will contain the formatted employee name.

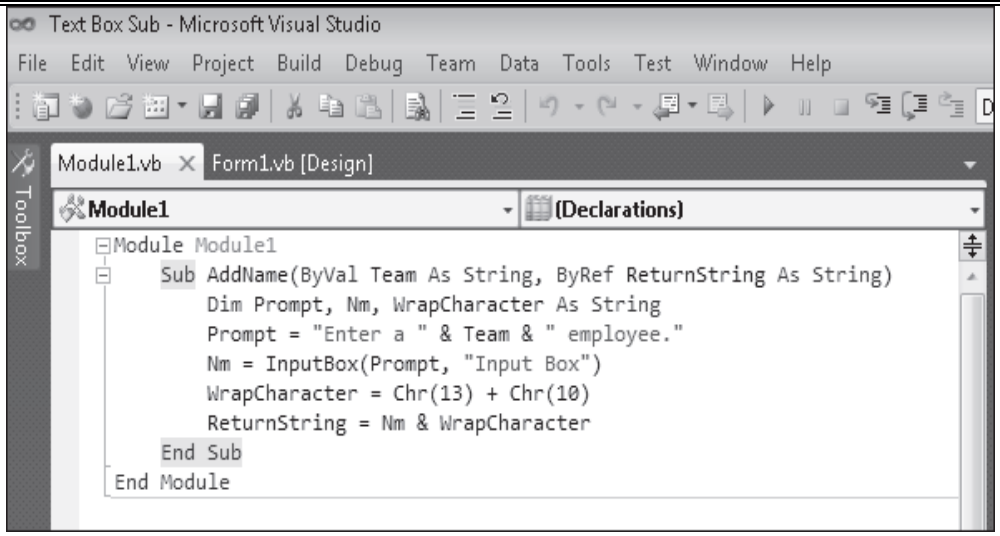
`ReturnString` is declared with the `ByRef` keyword so that any changes made to this argument in the procedure will be passed back to the calling routine through the argument.

Before the employee name is returned, carriage return and linefeed characters are appended to the string so that each name in the text box will appear on its own line. You can use this general technique in any string to create a new line.

`Chr(13)` = Carriage Return - (moves cursor to lefttmost side)
`Chr(10)` = New Line (drops cursor down one line)

The combination of both is used to start typing at the begining of a new line.

Your Code Editor looks like this:



1. Click Event Procedure of Buttons

i. Event Procedure for `btnSales_Click`

Type the following statements in the procedure:

```
Dim SalesPosition As String = ""
AddName("Sales", SalesPosition)
txtSales.Text = txtSales.Text & SalesPosition
```

The call to the `AddName` Sub procedure includes one argument passed by value (“`Sales`”) and one argument passed by reference (`SalesPosition`). The last line uses the argument passed by reference to add text to the `txtSales` text box. The concatenation operator (&) adds the new name to the end of the text in the text box.

ii. Event Procedure for `btnMkt_Click`

Type the following statements in the event procedure:

```
Dim MktPosition As String = ""
AddName("Marketing", MktPosition)
txtMkt.Text = txtMkt.Text & MktPosition
```

This event procedure is identical to `btnSales_Click`, except that it sends “Marketing” to the `AddName` procedure and updates the `txtMkt` text box. (The name of the local return variable `MktPosition` was renamed to make it more intuitive).

iii. Event Procedure for `btnQuit_Click`

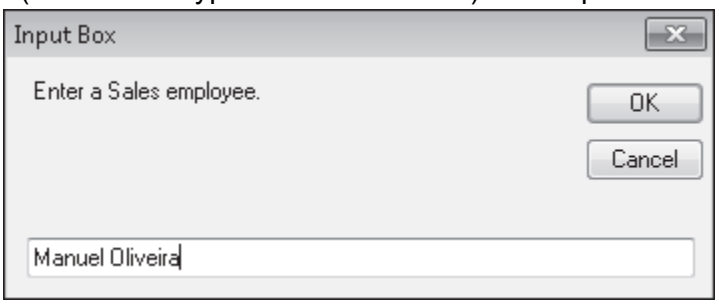
Type the following statements in the event procedure:

```
End
```

2. Run the Text Box Sub program

i. Click the Start Debugging button on the Standard toolbar to run the program.

ii. Click the Add Name button under the Sales text box, and then type `Manuel Oliveira` in the input box. (Feel free to type a different name.).Your input box looks like this:



iii. Click the OK button to add the name to the Sales text box. The name appears in the first text box.

iv. Click the Add Name button under the Marketing text box, type `Raymond Fong` in the Marketing input box, and then press ENTER. The name appears in the Marketing text box. Your screen looks like this:

v. Enter a few more names in each of the text boxes. This is your chance to create your own dream office staffing configurations. Each name appears on its own line in the text boxes. The text boxes don't scroll automatically, so you won't see every name you've entered if you enter more names than can fit in a text box. You can use the scroll bars to access names that aren't visible.

vi. When you've finished, click the Quit button to stop the program. You've demonstrated that one Sub procedure can manage input tasks from two or more event procedures. Using this basic concept as a starting point, you can now create more sophisticated programs that use Sub and Function procedures as organizing tools and that place common tasks in logical units that can be called over and over again.

Conlusion:

You can use segment of sub procedure code whenever you like, just by referring to it by name. Of course, you can have your code perform task of Adding Name more than one times. You can add as many as you like. And for whatever you like.