**ITEC397 Chapter 09 LABWORK**

**Example: Student scores input and letter grade calculation form**

1. **Create and design the UserForm**
* On the VBAEditor, Project Tree, with the right button of your mouse, select the menu options insert 🡪 UserForm.
* Add and place the Labels, textboxes and commandbuttons to the form as it is given in the design below:



* Rename the captions of the textboxes as: texbox\_stdno, textbox\_std\_name, …
* Rename the caption of the commandbuttons as; commandbutton\_save, commandbutton\_delete and commandbutton\_exit.
1. **Design the TabSheet 1 as the image given below**
* For keeping the data Range A3 to G31 will be used
* DO Insert 🡪 Draw, and select a rectangle shape. To the B1, draw a rectangle, double click it and enter the caption “Edit”.
* 



* To the second row, enter the column captions as they are given above.
1. **UserForm activation**
* We need a subroutine (sub program) to activate the UserForm, so on the VBAEditor, Project Tree, select Sheet 1 (Sheet 1) and with clicking right button of mouse select “View Code”.
* To the opened window add the following code:

*Sub Open\_form()*

 *UserForm1.Show*

 *UserForm1.TextBox\_stdno.SetFocus*

*End Sub*

* But, how this subroutine will be called? Select the rectangle you draw on B1, and click the right button of your mouse, and from the menu coming select the option “Assign Macro”. Then select the existing macro name Open\_form().
* Now, you turned this rectangle to a button which will activate the UserForm1 when it will be clicked.

1. **Configuring the TextBoxes**
* Set the MaxLengths of the TextBoxes. 10 characters to stdno, 25 characters to the names and 3 characters to the exam scores. Watch out, assigning three characters for an exam scores will allow the entry of a number like 999. In order to avoid this possibility, you have to control the input before exiting from the TextBoxes. Also leaving the TextBoxes of mt1, mt2, and final without entering any exam score should be not allowed. So, the following event subroutine has to be developed.

*Private Sub TextBox\_mt1\_Exit(ByVal Cancel As MSForms.ReturnBoolean)*

 *If (Not IsNumeric(TextBox\_mt1.Value)) Then*

 *MsgBox "Mt-1score has to be a numeric value", vbCritical*

 *TextBox\_mt1.Value = ""*

 *Cancel = True*

 *'TextBox\_std\_mt1.SetFocus*

 *Else*

 *If (TextBox\_mt1.Value > 100) Or (TextBox\_mt1.Value < 0) Then*

 *MsgBox "Mt-1score has to be a number betwee (0-100)", vbCritical*

 *TextBox\_mt1.Value = ""*

 *Cancel = True*

 *' TextBox\_mt1.SetFocus*

 *End If*

 *End If*

* The \_Exit event is giving you an argument “Cancel”, and you can disallow the control to exit from this subroutine, so from the TextBox with assigning “True” to that argument (Cancel).
* This subroutine will check the input and if it is not a numeric value (so, empty), or it is out of the range of legal exam scores (0-100), it will warn the user and will keep the control within the TextBox.
* Add the similar input control subroutines for the TextBoxes of mt2 and final exam scores.
1. **Configuring and Controlling the CommandButtons**
* In order to put Hot Keys to the CommandButtons, set the characters S, D, and E to the “Accelerator” properties of the CommandButtons of “Save”, “Delete”, and “Exit”.
* If you want to set Hot Keys to the TextBoxes that are actually not having such a property. The trick is that, you have to set the “Hot Key” character to the “Accelerator” property of the related “Label” and be sure that the “TabIndex” of the label has to be one less than the “TabIndex” of the related TextBox.
* In order to close the activated UseerForm add the following code to the \_Click() event of the Exit CommandButton.

*Private Sub CommandButton\_Exit\_Click()*

 *Unload Me*

*End Sub*

* To the \_Click() event of the “Save” button add the following code.

*Private Sub CommandButton\_save\_Click()*

*Total = 0*

*Ch1 = TextBox\_stdno.Text = ""*

*Ch2 = TextBox\_std\_name.Text = ""*

*Ch3 = TextBox\_std\_surname.Text = ""*

*Ch4 = TextBox\_mt1.Text = ""*

*Ch5 = TextBox\_mt2 = ""*

*Ch6 = TextBox\_final.Text = ""*

 *If (((((Ch1 Or Ch2) Or Ch3) Or Ch4) Or Ch5) Or Ch6) Then*

 *'Is Form is Empty?*

 *MsgBox "All fields have to be filled for saving!" & vbCrLf & " Please try again"*

 *Else*

 *If (ActiveCell.Value <> "") Then*

 *'MsgBox "Edit"*

 *Edit*

 *Else*

 *'MsgBox "Append"*

 *Append*

 *End If*

 *End If*

*End Sub*

* At the first part, we are trying to be sure that no TextBox is left as empty before saving the data the excel sheet.
* Then it is checked if the UserForm is called when the cursor is on an empty cell (So a new record will be appended to the excel sheet) or on a data (That implies, user wants to edit an existing row of data).
* With respect to the if condition explained above, “Edit”, or “Append” subroutines will be called.
1. **Initialize the UserForm1**
* UserForm1 will be activate to do a process on an existing line of student data (Edit, delete) or append a new student data to the list. So, on the activation of the UserForm1, we have to copy the data of the selected student from the list to the TextBoxes if the cursor is on a non-empty cell. In order to do these tasks, the following code has to be written in to the \_Activate() event of the UserForm1

*Private Sub UserForm\_Activate()*

 *n = 0*

 *If (ActiveCell.Value <> "") Then*

 *r = ActiveCell.Row*

 *c = "A"*

 *t = c & r*

 *Sheet1.Range(t).Select ‘Set the curser to the first cell of the selected row*

*‘Copying the data of the selected student into the TextBoxes*

 *TextBox\_stdno.Value = ActiveCell.Value*

 *TextBox\_std\_name.Value = ActiveCell.Offset(0, 1).Value*

 *TextBox\_std\_surname.Value = ActiveCell.Offset(0, 2).Value*

 *TextBox\_mt1.Value = ActiveCell.Offset(0, 3).Value*

 *TextBox\_mt2.Value = ActiveCell.Offset(0, 4).Value*

 *TextBox\_final.Value = ActiveCell.Offset(0, 5).Value*

 *End If*

*End Sub*

1. **Edit and Append processes**
* Complete the content of the subroutine Edit() with the following code.

*Private Sub Edit()*

 *r = ActiveCell.Row*

 *c = "A"*

 *t = c & r*

 *Sheet1.Range(t).Select ‘Set the cursor to the first cell of the selected row*

 *‘Replace the content of the cells with the updated values of TextBoxes*

 *ActiveCell = TextBox\_stdno.Value*

 *ActiveCell.Offset(0, 1) = TextBox\_std\_name.Value*

 *ActiveCell.Offset(0, 2) = TextBox\_std\_surname.Value*

 *ActiveCell.Offset(0, 3) = TextBox\_mt1.Value*

 *ActiveCell.Offset(0, 4) = TextBox\_mt2.Value*

 *ActiveCell.Offset(0, 5) = TextBox\_final.Value*

 *‘Calculate the total score of the student*

 *Total = 0.3 \* TextBox\_mt1.Value + \_*

 *0.3 \* TextBox\_mt2.Value + \_*

 *0.4 \* TextBox\_final.Value*

*‘ Calculate deserved the letter grade of the student*

*Select Case (Total)*

 *Case Is > 89*

 *ActiveCell.Offset(0, 6) = "A "*

 *Case Is > 79*

 *ActiveCell.Offset(0, 6) = "B "*

 *Case Is > 69*

 *ActiveCell.Offset(0, 6) = "C "*

 *Case Is > 59*

 *ActiveCell.Offset(0, 6) = "D "*

 *Case Is > 49*

 *ActiveCell.Offset(0, 6) = "E "*

 *Case Else*

 *ActiveCell.Offset(0, 6) = "F "*

*End Select*

*‘Empty the values of the TextBoxes*

 *TextBox\_stdno.Value = ""*

 *TextBox\_std\_name.Value = ""*

 *TextBox\_std\_surname.Value = ""*

 *TextBox\_mt1.Value = ""*

 *TextBox\_mt2.Value = ""*

 *TextBox\_final.Value = ""*

 *Sheet1.Range("H10").Select ‘Not necessary step*

 *TextBox\_stdno.SetFocus ‘Put the cursor into the TextBox\_stdno*

*End Sub*

* And complete the content of the subroutine Append with the given code below:

*Private Sub Append()*

*‘ Find and locate the cursor to the first cell of the empty row, coming just*

*‘ after the last data row*

 *Range("A2").Select*

 *Range("A1").End(xlDown).Offset(1 + row\_count(), 0).Select*

*‘ row\_count is a programmer defined function counting and return the number*

*‘ of already existing rows in the list*

*‘Copy the content of the TextBoxes to the data row on the list*

 *ActiveCell = TextBox\_stdno.Value*

 *ActiveCell.Offset(0, 1) = TextBox\_std\_name.Value*

 *ActiveCell.Offset(0, 2) = TextBox\_std\_surname.Value*

 *ActiveCell.Offset(0, 3) = TextBox\_mt1.Value*

 *ActiveCell.Offset(0, 4) = TextBox\_mt2.Value*

 *ActiveCell.Offset(0, 5) = TextBox\_final.Value*

 *Total = 0.3 \* TextBox\_mt1.Value + \_*

 *0.3 \* TextBox\_mt2.Value + \_*

 *0.4 \* TextBox\_final.Value*

*Select Case (Total)*

 *Case Is > 89*

 *ActiveCell.Offset(0, 6) = "A "*

 *Case Is > 79*

 *ActiveCell.Offset(0, 6) = "B "*

 *Case Is > 69*

 *ActiveCell.Offset(0, 6) = "C "*

 *Case Is > 59*

 *ActiveCell.Offset(0, 6) = "D "*

 *Case Is > 49*

 *ActiveCell.Offset(0, 6) = "E "*

 *Case Else*

 *ActiveCell.Offset(0, 6) = "F "*

*End Select*

 *TextBox\_stdno.Value = ""*

 *TextBox\_std\_name.Value = ""*

 *TextBox\_std\_surname.Value = ""*

 *TextBox\_mt1.Value = ""*

 *TextBox\_mt2.Value = ""*

 *TextBox\_final.Value = ""*

 *Sheet1.Range("H10").Select ‘ Unnecessary step of work*

 *TextBox\_stdno.SetFocus*

*End Sub*

* Write the code of the programmer defined function row\_count(), that is used in append subroutine for deciding the position of the first empty row coming after the last data row.

*Private Function row\_count()*

 *row\_count = 0*

 *For n = 3 To 31 ‘The rows of the list was defined to be between 3 to 31*

 *t = "A" & n*

 *If Range(t).Value <> "" Then ‘from A3 to A31 non-empty rows will be count*

 *row\_count = row\_count + 1*

 *End If*

 *Next n*

*End Function*

1. **Deleting a row processes**
* Complete the content of the \_click() event of the COmmandButton\_Exit with the following code.

*Private Sub CommandButton\_delete\_Click()*

 *‘If the selected cell is not within the valid range or it is on an empty cell*

 *‘ Do not delete, close the UserForm*

 *If (ActiveCell.Value = "") Or (ActiveCell.Row < 3) Then*

 *MsgBox "You did not select any row for deleting!"*

 *TextBox\_stdno.Text = ""*

 *TextBox\_std\_name.Text = ""*

 *TextBox\_std\_surname.Text = ""*

 *TextBox\_mt1.Text = ""*

 *TextBox\_mt2 = ""*

 *TextBox\_final.Text = ""*

 *Unload Me*

 *End If*

 *‘Decide the position of the row that will be deleted*

 *r = ActiveCell.Row ‘The row that will be deleted*

 *nr = row\_count() + 2 ‘Last non-empty row in the list*

 *If ((r - 2) <> nr) Then ‘If not the last row is selected for deleting*

 *‘Every row on the list will be shifted to the previous row position*

 *For m = r To nr – 1*

 *n = "A" & m ‘Current row position*

 *o = "A" & (m + 1) ‘ Next row position*

 *Range(n).Value = Range(o).Value*

 *Range(n).Offset(0, 1).Value = Range(o).Offset(0, 1).Value*

 *Range(n).Offset(0, 2).Value = Range(o).Offset(0, 2).Value*

 *Range(n).Offset(0, 3).Value = Range(o).Offset(0, 3).Value*

 *Range(n).Offset(0, 4).Value = Range(o).Offset(0, 4).Value*

 *Range(n).Offset(0, 5).Value = Range(o).Offset(0, 5).Value*

 *Range(n).Offset(0, 6).Value = Range(o).Offset(0, 6).Value*

 *Next m*

 *n = "A" & nr ‘ The Last Row position in the list*

 *Else*

 *n = "A" & r ‘The selected last row position*

 *End If*

 *‘ Get empty this row*

 *Range(n).Value=””*

 *Range(n).Offset(0, 1).Value = ""*

 *Range(n).Offset(0, 2).Value = ""*

 *Range(n).Offset(0, 3).Value = ""*

 *Range(n).Offset(0, 4).Value = ""*

 *Range(n).Offset(0, 5).Value = ""*

 *Range(n).Offset(0, 6).Value = ""*

 *Unload Me ‘ Close the UserForm1*

*End Sub*

1. **Makeup and protection of the working place**
* After you may be sure your application is working properly, you may need to give a more professional look to your application, and also you may need to protect the working environment from any user mistake. This last touches are going to be for this respect.
* For disabling the grids follow the following menu items: File 🡪 Options 🡪Advance

Find the “Displays Options For This Sheet” : Sheet1, and un-check the option “show gridlines”, and “row and column headers”, then click OK.

* Now, we will protect our work from user mistakes. From the “File” menu, select the “Protect Workbook” option. Then, select the “Protect Current Sheet” option.



* Just enabled the first two options and keep the rest as disabled. Enter a password, for this example I set the password as “1”, then click OK. The password is going to re-asked from you, again enter “1” and continue.
* Now, the whole worksheet is protected for any change, and even your VBA code will not allowed to do any change on this worksheet.
* In order to allow your VBA code to do any changes on this worksheet we have to update as it will be:

*Private Sub UserForm\_Activate()*

 *n = 0*

 *If (ActiveCell.Value <> "") Then*

 *r = ActiveCell.Row*

 *c = "A"*

 *t = c & r*

 *Sheet1.Range(t).Select ‘Set the curser to the first cell of the selected row*

*‘Copying the data of the selected student into the TextBoxes*

 *TextBox\_stdno.Value = ActiveCell.Value*

 *TextBox\_std\_name.Value = ActiveCell.Offset(0, 1).Value*

 *TextBox\_std\_surname.Value = ActiveCell.Offset(0, 2).Value*

 *TextBox\_mt1.Value = ActiveCell.Offset(0, 3).Value*

 *TextBox\_mt2.Value = ActiveCell.Offset(0, 4).Value*

 *TextBox\_final.Value = ActiveCell.Offset(0, 5).Value*

 *End If*

 *With Worksheets("Sheet1")*

 *.EnableAutoFilter = True*

 *.Protect Password:="1", \_*

 *Contents:=True, DrawingObjects:=False, UserInterfaceOnly:=True, \_*

 *AllowFormattingCells:=True*

 *End With*

*End Sub*

* The last part will be used for accessing the worksheet and temporarily open the protection for your VBA code. Watch out the line .*Protect Password:="1",* it is using the password that you set to protect-unprotect the worksheet on the previous step.
* Of course, in a real application, you have to decide a stronger password that you will remember and will be hard to be guessed or found by user tries.