**Lab Workshop #6**

Purpose: use the macro recorder to create an automated procedure

 modify and clean up VBA code using the Visual Basic Editor

 create two VBA "Sub" procedures from scratch and test them

 create a VBA user-defined "Function" from scratch and test it

1. Launch Excel 2007. Click the **Office** button, **Excel Options** button, and, with **Popular** selected, check the box **Show the Developer tab in the ribbon**. Click **OK**.

2. You may find it useful to refer to the notes from **Class10**. You can follow the steps starting on slide 12 for recording a macro.

 Starting with a blank workbook, use the macro recorder to create a VBA "Sub" procedure called "SpecialGreen", with shortcut key, *Ctrl-Shift-G*, attached to ***This Workbook***, that formats a cell with the following characteristics:

* a light green background color
* font: bold, italic, 10-pt, arial
* bold, single border around cell
* alignment: horizontally centered
* number format: 3 places to the right of the decimal point displayed

 Test the macro on another cell to make sure it works. Clean up the VBA code using the Visual Basic Editor, removing any statements that are unnecessary. In the comment area of the VBA code, add a comment, starting with an apostrophe ( **'** ), with your name, lab section and the date.

 Test the macro again on a selection of several cells. Save the workbook as **lab6a.xlsx** and close it.

3. Open a new workbook.

 You will now create a Sub that illustrates how information can be exchanged between cells on the spreadsheet and variables in the VBA procedure.

 Start in the spreadsheet and enter the following values in the cells shown:

 

 Switch to the VBE ( *Alt-F11* ) and, with the current project selected in the Project Explorer window, select Insert → Module from the menu. Enter the following VBA code:

 

 The purpose of this Sub is to switch the values in cells A1 and A2. It's necessary to use a variable in VBA for temporary storage of the cell A1 value. The switch takes place as shown in the diagram on the next page.



 The three statements in the program are represented by the numbers 1, 2 and 3 in the diagram.

 Switch back to the spreadsheet and run the procedure. You can do this from the Developer ribbon, Code frame and Macros button, or *Alt-F8* shortcut. In the Macro dialog box, double-click on the SwapValues entry to run it. You should see the values in cells A1 and A2 switch. Run the macro again and they should switch back.

 Using Insert → Form Controls on the Developer ribbon, add a rectangular button (  ) to the spreadsheet that runs the macro. Change the button label to "Switch." Add your name and the date to the spreadsheet in cells B1 and B2.

 Demonstrate your macro to your TA and have him/her initial here.

 Save the workbook as **lab6b.xlsx** and close it out.

4. Open a new workbook. In this workbook, you will create a user-defined function (UDF). User-defined functions are like the built-in functions in Excel (SQRT, EXP, LN, VLOOKUP, IF, etc.) except that you create them. The ability to create your own functions really expands your horizon with Excel.

 You will create a function here that implements the following calculation:

  We'll call this function SgnSqr(•) in Excel.

 Open the VBE and insert a new module (Insert → Module). You should take care to note, in the Project Explorer window, that your current workbook's project is selected before you insert the module. In the blank code window, type in the following VBA code:

 

 Switch back to the spreadsheet to try the function out. Enter a value of 2 in cell B2. In cell C2 enter the formula:

 **=SgnSqr(B2)**

 What is displayed in cell C2? \_\_\_\_\_\_\_

 Change the value in cell B2 to **–2**. What now is displayed in cell C2? \_\_\_\_\_\_\_\_\_

 Switch back to the VBE window. Add a comment ( starting with an **'** ) just below the Function statement with your name and the date.

 There are some features of this Function that should be pointed out, even though we'll cover all this in detail a bit later.

 1) a UDF starts with the **Function** statement and finishes with the **End Function** statement[[1]](#footnote-1)

 2) the SgnSqr function has a *formal argument*, **x**, also called a "dummy" argument

 3) the VBA code makes use of an **If ... Then ... Else ... End If** statement structure to implement the two-way design of the function

 4) the result of the function's calculation, in either branch of the If statement, is assigned equal to the name of the function, **SgnSqr**.

 Save this workbook as **lab6c.xlsx** and close it out.

5. Leave Excel and return to Windows.

**End of Lab Workshop #6**

1. You've noticed that the bracketing End Sub or End Function statements are inserted automatically by the VBE after you type in the Sub or Function statements. [↑](#footnote-ref-1)