Wednesday, September 5th handout. To be used on 9/5 and 9/7 in lab and discussed during the week #3 lectures in Wright 5. (Note the classroom change to WRIGHT 5). WRIGHT 5 WRIGHT 5

VIP: You do NOT have to understand nor are you expected to understand all or very much of the Excel or the VBA and macros concepts in this hands-on project until sometime in WEEK #4 or WEEK #5. This is week #2.

During week #3, on either Monday, September 10th or Wednesday, September 12th, an assignment will be given that is similar to this project, but simpler.

On the Z: drive, we will have a folder named Excel022 and in that folder there will be two files: Z:\Excel022 contains:

Ord9711.txt **Thursday, September 22nd, 2011**
Orders.dbf ***StudioIT 2 ITT 136 lab class***

MAP NETWORK DRIVE * Map letter P: to \\PLEIADES\COURSES WRTHALL WRIGHT

We will get these two files from the P:\Jacobson folder.

If you do NOT have the P: drive network connection:

1. Right click on the My Computer desktop icon.

2. Choose Map Network Drive from the list of popupmenu commands.

3. Select P: as the Drive letter.

4. Type in the following for the Path: \\Pleiades\Courses (\ is backslash, not /, but \)

5. For the username, type WRTHALL and when asked for password, type WRIGHT to be connected to \\PLEIADES\COURSES.

6. Make the Excel022 folder, copy the ord9711.txt and orders.dbf files over to your Excel022 folder. ----------- ----------

7. Open a new workbook. Save it with the name Lesson2Sep5 or Lesson2Sep5.xls in the Excel022 folder on your Z: drive.

Here is how the macro ImportFile might appear after we modify it so that it uses the GetOpenFilename method of the Excel Application object. It consists of 6 statements.

Note:

```
Range("A2").Select
Selection.EntireRow.Delete
```

could have been recorded instead of:

```
Rows("2:2").Select
Selection.Delete Shift:=xlUp
```

```
Sub ImportFile()
'
' ImportFile Macro Macro recorded 9/5/2001 by Mark F Jacobson

    myFile = Application.GetOpenFilename("Text Files,*.txt")

    Workbooks.OpenText Filename:=myFile, Origin:= _
        xlWindows, StartRow:=4, DataType:=xlFixedWidth, FieldInfo:= _
        Array(Array(0, 1), Array(8, 1), Array(20, 1), Array(26, _
        1), Array(41, 1), Array(49, 1), Array(59, 1), Array _
        (67, 1))

    ActiveSheet.Move Before:=Workbooks("Lesson2Sep5th.xls").Sheets(1)

    Rows("2:2").Select

    Selection.Delete Shift:=xlUp

    Range("A1").Select
End Sub
```

Here is how it looks after adding line continuations, which are created by typing a space followed by an underscore character.

Only the Workbooks.OpenText method statement has been modified by formatting it so it is more readable than the macro recorder made it.

Without the " _" pair of symbols, VBA would give you an error message.

```
Sub ImportFile()
'
' ImportFile Macro Macro recorded 9/5/2001 by Mark F Jacobson

    myFile = Application.GetOpenFilename("Text Files,*.txt")

    Workbooks.OpenText Filename:=myFile, _
                       Origin:= xlWindows, StartRow:=4, _
                       DataType:=xlFixedWidth, _
                       FieldInfo:=Array(Array(0, 1), Array(8, 1), _
                                  Array(20, 1), Array(26, 1), _
                                  Array(41, 1), Array(49, 1), _
                                  Array(59, 1), Array(67, 1))

    ActiveSheet.Move Before:=Workbooks("Lesson2Sep5th.xls").Sheets(1)
    Rows("2:2").Select
    Selection.Delete Shift:=xlUp
    Range("A1").Select
End Sub
```

Note that the above macro consists of only 6 different statements.

One of the statements has 6 different line continuations " _",
   and is spread out over 7 lines.

Before reformatting, it had only 3 different line continuations
    and was spread out over only 4 lines.

The macro originally only had 5 useful statements, but we added a new 1st statement
    [myFile = Application.GetOpenFilename("Text Files,*.txt")]
    and then we modifed the Workbooks.OpenText statement
    to use the myFile variable.

    We may have deleted a few statements that resized the workbook or application
    windows or that changed the directory or drive.

The following is the FillLabels macro. See the Watch the FillLabels macro run
               portion of the handout you got in Lab #1 on Friday, August 31st.

```
Sub FillLabels()
'
' FillLabels Recorded 9/5/2001 by Mark F Jacobson

    Range("A1").Select
    Selection.CurrentRegion.Select
    Selection.SpecialCells(xlCellTypeBlanks).Select
    Selection.FormulaR1C1 = "=R[-1]C"
    Selection.CurrentRegion.Select

    Selection.Copy
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
                           SkipBlanks:=False, Transpose:=False
    Application.CutCopyMode = False

    Range("A1").Select
End Sub
```

What does the statement [Selection.FormulaR1C1 = "=R[-1]C"] mean?

How could we record that differently so that it said
   ActiveCell.FormulaR1C1 = instead of Selection.FormulaR1C1 =?

   Holding down the Ctrl key (Control key) while pressing Enter causes
   the formula to be used in all the cells of the Selection, whereas
   just pressing Enter by itself means only the ActiveCell object is
   effected and receives the new formula or value.

How could we record it differently so that it said
   Selection.FormulaR1C1 = "=R2C4"?

What function key would be involved in making this recording simpler?
    The F4 key enables you to easily cycle through absolute, mixed and
    relative cell references.

    References are either: relative, such as D4
                            absolute, such as D4
                            mixed, such as $D4   <--- column absolute
                                        or D$4   <--- row absolute

---

```
Sub AddDates()
'
' AddDates Macro recorded 9/5/2001 by Mark F Jacobson

    Range("A1").Select
    Selection.EntireColumn.Insert

    ActiveCell.FormulaR1C1 = "Date"            ' Label typed, followed by Enter

    Range("A2").Select
    Selection.CurrentRegion.Select
    Selection.SpecialCells(xlCellTypeBlanks).Select

    Selection.FormulaR1C1 = "9/1/2001"         ' Sep-01 followed by Ctrl+Enter
    Range("A1").Select
End Sub

1.  Add the following statement to the recorded code above, just before the

        ActiveCell.FormulaR1C1 = "Date" statement:

        myDate = InputBox("Enter the date in MMM-YY format")

2.  Change the ActiveCell.FormulaR1C1 = "Date"

            to ActiveCell.FormulaR1C1 = myDate

3.  Note the two different statements that use the FormulaR1C1 property,

       but it is either ActiveCell.FormulaR1C1 =         <--- Enter

                    or Selection.FormulaR1C1 =          <--- Ctrl+Enter

--------------------------------------------------------------------------------

Sub AppendDatabase()
'
' AppendDatabase Macro recorded 9/5/2001 by Mark F Jacobson

    Range("A1").Select
    Selection.EntireRow.Delete
    Selection.CurrentRegion.Select
    Selection.Copy
    Workbooks.Open Filename:="Z:\Excel022\Orders.dbf"
    Selection.End(xlDown).Select

    Range("A3301").Select

    ActiveSheet.Paste
    Application.CutCopyMode = False
    Selection.CurrentRegion.Select

    ActiveWorkbook.Names.Add Name:="Database", _
                             RefersToR1C1:="=Orders!R1C1:R3478C9"
    ActiveWorkbook.Close
    Range("A1").Select
End Sub
```

*** Change #1:

Statement 7, Range("A3301").Select is wrong and will not work next month.

We will record a macro using relative references to see what it needs to be replaced with. The Ctrl+Down Arrow works fine to get us to the last row, whether it be row 3300 or 3478 or 245 or 3521 or whatever.

Here is the result of recording a macro with the Relative References button down (clicked to switch from Absolute References to Relative recording). The DOWN ARROW key was clicked once to record a relative movement to the next row, but staying in the same column. The macro was named RelativeMove1RowDown and can be deleted later on, after we see the syntax for the VBA statement that we need.

```
Sub RelativeMove1RowDown()
'
' RelativeMove1RowDown recorded 9/5/2001 by Mark F Jacobson
'
    ActiveCell.Offset(1, 0).Range("A1").Select
End Sub
```

Replace the statement: `Range("A3301").Select`
with: `ActiveCell.Offset(1, 0).Range("A1").Select`

*** Change #2:

Replace the entire statement:

```
ActiveWorkbook.Names.Add Name:="Database", _
    RefersToR1C1:="=Orders!R1C1:R3478C9"
```

with:

```
Selection.Name = "Database"
```

Name is a property of a range in Excel.
All we have to do to name a range is to assign a word in quotation marks to be the value of the Name property for the specified range, which in this case is the Selection object, obtained by Ctrl+Shift+* and thus the entire set of rows and columns now in the updated database of sales information.

*** Change #3

Here is the statement that closes the workbook: `ActiveWorkbook.Close`

Change it to this: `ActiveWorkbook.Close SaveChanges:=False`

This will end the automatic triggering of the prompt that asks you if you want to save changes to the file.

Finally, we will record a macro to delete the imported text file worksheet. It is not needed anymore, after we have appended all of the information to the database file.

```
Sub DeleteSheet()
    ActiveWindow.SelectedSheets.Delete
End Sub
```

After recording this macro, we will add a statement to it so that it runs quietly and does not display the warning message.

Application.DisplayAlerts = False is the statement that does the job

```
Sub DeleteSheet()
    Application.DisplayAlerts = False        ' <--- Now it runs quietly
    ActiveWindow.SelectedSheets.Delete
End Sub
```

Now all of the pieces of the project have a macro that is tailored to speed up the process of carrying out the monthly project update.

But the user of your Excel macros does not need to run the 5 separate macros each month. They can all be put together into one macro. We will record a macro that runs the 5 different macros in the correct sequence.

1. Click the Record Macro button on the Visual Basic toolbar, or use the Tools menu, Macros, Record macro command.

2. Type MonthlyProject as the macro name. Click OK to start recording it.

3. Click the Run Macro button on the Visual Basic toolbar.
   Select ImportFile (click on it) and then click the Run button.

4. Select the text file you want to import, which is ord9711.txt, and then click the Open button. Here is the statement you are responding to:

   ```
   myFile = Application.GetOpenFilename("Text Files,*.txt")
   ```

5. Click the Run macro button, click FillLabels, and click Run.

6. Click Run macro button, click AddDates, and then click Run.

7. Type a date with mmm-yy format, such as Jul-01 or Sep-01 or Dec-89 and then click the OK button. You are responding to this InputBox statement

   ```
   myDate = InputBox("Enter the date in MMM-YY format")
   ```

8. Click the Run macro button, click AppendDatabase, and click Run.

9. Click Run macro button, click DeleteSheet, and then click Run.

10. Click the Stop Recording button.

Here is what the MonthlyProject macro should now look like:

```
Sub MonthlyProject()  '  Recorded 9/5/2001 by Mark Jacobson

    Application.Run "Lesson2Sep5th.xls!ImportFile"
    Application.Run "Lesson2Sep5th.xls!FillLabels"
    Application.Run "Lesson2Sep5th.xls!AddDates"
    Application.Run "Lesson2Sep5th.xls!AppendDatabase"
    Application.Run "Lesson2Sep5th.xls!DeleteSheet"

End Sub
```

Forget the VBA and Excel macros skills you have begun to develop from the above series of macros. How many new things did you learn about Excel?

1. **Did you know about naming ranges?**
2. **Did you know techniques for selecting or concept of a Current Region?**
3. **Did you know how to select just the blanks?**
4. **Did you know about Paste Special?**
5. **Importing text files?**
6. **Absolute, Relative and Mixed cell references? The F4 key?**
7. **Ctrl+DownArrow to move to the last row of the current region?**
8. **How to move a sheet from one workbook to another?**
9. **How to apply a label or formula or value to an entire region of cells instead of just one cell (the active cell)?**
10. **How to Insert or Delete entire rows or columns?**
11. **Importing a database file into Excel?**
12. **Appending information to an Excel range and exporting the new information to a database file?**