'-----------------------------------------------------------------------------------------------------------------------------------

'Last Update: Oct 25, 2013

'-----------------------------------------------------------------------------------------------------------------------------------

Option Explicit

Option Base 1

'-----------------------------------------------------------------------------------------------------------------------------------

'-----------------------------------------------------------------------------------------------------------------------------------

'Lesson 3.9.1: Math Class Functions

'-----------------------------------------------------------------------------------------------------------------------------------

'-----------------------------------------------------------------------------------------------------------------------------------

'If you have created a model in Excel, you should be well-versed in prebuilt mathematical functions. VBA too has numerous built-in

'mathematical functions that we can call in our programs. The following table summarizes the available functions found in the Math

'namespace that may be important in quantitative analysis. To call these functions, we need to precede the function name with the

'class name Math and a dot (.). That is, the fully qualified function name for the Max() function, for example, is Math.Max().

Sub TEST\_MATH\_CLASS()

Debug.Print Math.Abs(-32) 'Abs: Returns the absolute value of x.

Debug.Print Math.Exp(0.034 \* 2) 'Exp: Returns e (the base of natural logarithms) raised to the power of x.

Debug.Print Log(23.54 / 23.45) 'Log: Returns the natural logarithm of x.

Debug.Print Sgn(-0.02) 'Sgn: Returns 1 if x is greater than 0; 0 if x equals 0; -1 if x is less than 0.

'Very Useful for Buy/Sell Signals!

Debug.Print Sqr(144) 'Sqr: Returns the square root of x.

'Please notice that VBA does not have a built-in Max/Min/Ceiling/FLoor functions. You can either use the Excel Built-In Functions

'from VBA, or code your own version.

Debug.Print WorksheetFunction.Max(43.34, 43.33) 'Max: Returns the maximum of two input arguments.

Debug.Print WorksheetFunction.Min(43.34, 43.33) 'Min: Returns the minimum of two input arguments.

Debug.Print WorksheetFunction.Ceiling(3444.343, 2) 'Ceiling: Returns the integer greater than or equal to the input argument.

Debug.Print WorksheetFunction.Floor(3444.343, 2) 'Floor: Returns the integer less than or equal to the input argument.

End Sub

'-----------------------------------------------------------------------------------------------------------------------------------

'-----------------------------------------------------------------------------------------------------------------------------------

'Lesson 3.9.2: Evaluation of formulas in text

'-----------------------------------------------------------------------------------------------------------------------------------

'-----------------------------------------------------------------------------------------------------------------------------------

'It would often be convenient to evaluate a function entered as text; for instance if we have the function for the deflection of a

'cantilever under point loading at the end: F\*L^3/(3\*E\*I) then it would be convenient to be able to allocate different values to

'F, L, E and I, and calculate the value of the function, without having to re-enter it.

'Unfortunately Excel provides no such function. Fortunately it is quite easy to write one. VBA has an Evaluate function that

'returns the numerical value of any text string representing a mathematical formula, but this is

'not directly available from the spreadsheet (see http://www.vertex42.com/ExcelArticles/evaluate-function.html).

'Below is a User Defined Function (UDF) to provide this functionality, together with examples of its application.

Function Eval(Func As String, \_

ParamA As Variant, \_

ValueA As Variant, \_

Optional ReturnType As Long = 1, \_

Optional CommaDec As Boolean = False) As Variant

Dim i As Long

Dim Eform As String

' Evaluate a function (Func), replacing the parameters listed in the

' range "ParamA" with the corresponding values in "ValueA"

'Func is a single cell or text string containing the function to be evaluated

'ParamA and ValueA are single column or single row ranges containing the same number of

'cells

' ReturnType = 0 to return the function with values substituted for variables

' ReturnType = 1 (default) to return evaluated function

' CommaDec = True to convert commas to decimal point and semi-colons to commas

' CommaDec = False (default) for no convertion.

GetArray ParamA

GetArray ValueA

Eform = Func

For i = 1 To UBound(ParamA, 1) - LBound(ParamA, 1) + 1

Eform = Replace(Eform, ParamA(i, 1), ValueA(i, 1))

Next i

'You may have a query about using the Eval User Defined Function (UDF) when the decimal separator is

'set to a comma, rather than a decimal point. In response to that there is the option CommaDec = True

'to the code that would replace all occurrences of Ò,Ó with Ò.Ó and Ò;Ó with Ò,Ó.

'This allows the Excel Evaluate command to operate correctly, regardless of how the values are entered.

'Also notice that Excel has option that allows the functions to work, without substituting commas

'for decimal points. In Excel 2010, under File-Options-Advanced, deselect ÒUse System SeparatorsÓ and

'enter a Ò,Ó for Decimal Separator, and a space for Thousands Separator.

'The spreadsheet will now display decimal numbers with a comma separator, and will only accept a comma

'as the separator for new numbers, but the Eval functions will work correctly.

'However Ð this (strangely) only seems to work if the region language is set to English. If the region is

'set to French (or presumably any other region that uses the comma separator) the original Eval functions

'no longer work. I have only given this limited testing (since I normally use the decimal point), so if

'anyone else has any comment on how the regional separator options work in Excel, please send me an email!

If CommaDec = True Then

' Replace all , with . and ; with ,

Eform = Replace(Eform, ",", ".")

Eform = Replace(Eform, ";", ",")

End If

'Unlike the Eval function, where parameter values are read from ranges on the worksheet,

'when the Evaluate function is used in a name definition the parameters must either be

'converted to numerical values, or must be defined as named ranges. Playing with this it

'occurred to me that it would be useful for Eval UDF to have the option to return a text

'string with the function parameters converted to values, rather than the evaluated function

'value.

If ReturnType = 0 Then

Eval = Eform

Else

Eval = Evaluate(Eform)

End If

Exit Function

GetArrayLine:

Return

End Function

Public Sub GetArray(arrayname) 'ByRef!!!!

Dim temp As Variant

'Traps to watch out:

'If a function may be used as a UDF or called from another VBA routine, you need to check if the data being

'passed is a worksheet range or if it is already an array.

'If the function parameter is a single cell range it will not be converted into an array.

'If the parameter is a single row range it will be converted to a 2D array with a single row. A 1D array

'therefore needs to be converted into the same form for consistency.

'Similarly, a single cell range, if converted into an array, will be converted into a 1D, base zero array.

'For consistency it needs to be converted into a 2D, base 1 array, with 1 row and 1 column.

If TypeName(arrayname) = "Range" Then

If arrayname.Rows.Count = 1 Then

If arrayname.Columns.Count = 1 Then

arrayname = Array(arrayname.Value2)

temp = arrayname(0)

ReDim arrayname(1 To 1, 1 To 1)

arrayname(1, 1) = temp

Else

arrayname = Array(arrayname.Value2)

arrayname = WorksheetFunction.Transpose(arrayname)

End If

Else

arrayname = arrayname.Value2

End If

ElseIf Not IsArray(arrayname) Then

arrayname = Array(arrayname)

arrayname = WorksheetFunction.Transpose(arrayname)

Else

arrayname = WorksheetFunction.Transpose(arrayname)

End If

End Sub

'SubstituteA, that works the same as the built in Substitute function, but on a range of values,

'rather than just 1. This in effect does the same as the Eval UDF when set to return a text

'string.

Function SubstituteA(Func As String, RepA As Variant, ValueA As Variant) As Variant

Dim Eform As String, i As Long, NumRep As Long

'SubstituteA returns the same string as Eval with a ReturnType of 0.

'The same results are also given by repeated application of the Excel built-in Substitute() function.

' Replace the parameters listed in the

' range "RepA" with the corresponding values in "ValueA", in the string "Func"

'Func is a single cell or text string.

'RepA and ValueA are single column or single row ranges containing the same number of

'cells

GetArray RepA

NumRep = UBound(RepA)

GetArray ValueA

Eform = Func

For i = 1 To NumRep

Eform = Replace(Eform, RepA(i, 1), ValueA(i, 1))

Next i

SubstituteA = Eform

End Function

'EvalText evaluates a text function consisting of numerical values, for example the output of the SubstituteA function.

'The text may also be evaluated using a defined name (see Eqn1\_1 in Name Manager).

Function EvalText(Func As String) As Variant

EvalText = Evaluate(Func)

End Function

Sub Evaluating\_text\_formulas() 'subroutine

Dim i As Long

Dim j As Long

Dim DST\_RNG As Range

Dim ADDRESS\_STR As String

Dim FORMULA\_STR As String

Dim FORMULAS\_STR As String

Dim SRC\_WSHEET As Worksheet

Set SRC\_WSHEET = Worksheets("WSHEET\_TEST")

On Error Resume Next

With SRC\_WSHEET.Parent

.Names("Eqn1\_1").Delete

.Names.Add Name:="Eqn1\_1", RefersToR1C1:="=EVALUATE(R6C10)"

.Names("Eqn2\_2").Delete

.Names.Add Name:="Eqn2\_2", RefersToR1C1:="=EVALUATE(R53C6)"

End With

On Error GoTo 0

With SRC\_WSHEET

With .Cells

.Clear

.ColumnWidth = 10

End With

FORMULAS\_STR = "$A$1|Function|$D$1|Result|$E$1|Display parameter values using Eval() UDF|$J$1|Display parameter values using Substitute() Function|" & \_

"$A$2|F\*L^3/(3\*E\*I)|$D$2|=eval(A2,A4:A7,B4:B7,,FALSE)|$E$2|=eval(A2,A4:A7,B4:B7,0)|$J$2|'=F\*L^3/(3\*E\*I)|$E$3|Evaluate usung EvalText() UDF|$J$3|" & \_

"=SUBSTITUTE(J2,A4,B4)|$A$4|F|$B$4|0.1|$E$4|=EvalText(E2)|$J$4|=SUBSTITUTE(J3,A5,B5)|$A$5|L|$B$5|10|$J$5|=SUBSTITUTE(J4,A6,B6)|$A$6|E|$B$6|200000|" & \_

"$J$6|=SUBSTITUTE(J5,A7,B7)|$A$7|I|$B$7|=1/12|$J$7|Evaluate using defined name|$J$8|=Eqn1\_1|$K$8| Eqn1\_1 is defined as: =EVALUATE($J$6)|" & \_

"$A$9|Function|$D$9|Result|$E$9|Display parameter values using SubstituteA() UDF|$A$10|'=Beta\*E\*K\*sin(I)|$D$10|=eval(A10,$A$18:$A$23,$H$18:$H$23)|" & \_

"$E$10|=substitutea(A10,A18:A23,H18:H23)|$E$11|Evaluate usung EvalText() UDF|$A$12|t|$B$12|0.1|$E$12|=EvalText(E10)|$A$13|b\_0|$B$13|3.9|$A$14|b\_1|" & \_

"$B$14|1.9|$A$15|h|$B$15|3.9|$A$16|E|$B$16|30000000|$A$17|PR|$B$17|0.285|$A$18|E|$H$18|=+B16|$A$19|K|$B$19|'=+t^3/3\*(2\*b\_0+4\*b\_1)|$H$19|" & \_

"=eval(B19,$A$12:$A$15,$B$12:$B$15)|$A$20|Cw|$B$20|+t\*b\_0^2/24\*(8\*b\_1^3+6\*h^2\*b\_1+h^2\*b\_0+12\*b\_1^2\*h)|$H$20|=eval(B20,$A$12:$A$15,$B$12:$B$15)|" & \_

"$A$21|G|$B$21|+E/(2\*(1+PR))|$H$21|=eval(B21,$A$12:$A$17,$B$12:$B$17)|$A$22|I|$H$22|=PI()/4|$A$23|Beta|$B$23|'=+(K\*G/(Cw\*E))^0.5|$H$23|" & \_

"=eval(B23,A18:A21,$H$18:$H$21)|$A$26|Using Eval with other UDFs|$A$28|Tfunc is a UDF evaluating: A \* Sin(x) ^ C + B \* Cos(x) ^ C|$A$30|" & \_

"'=B\*tfunc(A,B,C,x)|$C$30|=eval(A30,B32:E32,B33:E33)|$B$32|x|$C$32|A|$D$32|B|$E$32|C|$B$33|0.4|$C$33|2|$D$33|2|$E$33|3|$A$35|SubstituteA Function|" & \_

"$A$37|Function|$D$37|Result|$A$38|F\*L^3/(3\*E\*I)|$D$38|=eval(A38,A40:A43,B40:B43)|$A$40|F|$B$40|2.5|$A$41|L|$B$41|10|$A$42|E|$B$42|200000|" & \_

"$A$43|I|$B$43|=1/12|$A$46|SubstituteA Function|$F$46|=substitutea($A$38,$A$40:$A$43,$B$40:$B$43)|$A$47|Eval function with optional argument = 0|" & \_

"$F$47|=eval($A$38,$A$40:$A$43,$B$40:$B$43,0)|$A$49|Repeated use of built-in SUBSTITUTE() function|$F$49|'=F\*L^3/(3\*E\*I)|$F$50|" & \_

"=SUBSTITUTE(F49,A40,B40)|$F$51|=SUBSTITUTE(F50,A41,B41)|$F$52|=SUBSTITUTE(F51,A42,B42)|$F$53|=SUBSTITUTE(F52,A43,B43)|" & \_

"$A$54|Evaluation using EvalText() function|$F$54|=EvalText(F53)|$A$55|Evaluation using a defined name|$F$55|=Eqn2\_2|$G$55|" & \_

"Eqn2\_2 is defined as: =EVALUATE($F$53)|"

i = 1

Do

j = InStr(i, FORMULAS\_STR, "|")

If j = 0 Then: Exit Do

ADDRESS\_STR = Mid(FORMULAS\_STR, i, j - i)

i = j + 1

j = InStr(i, FORMULAS\_STR, "|")

FORMULA\_STR = Mid(FORMULAS\_STR, i, j - i)

If Left(FORMULA\_STR, 1) = "=" Then

.Range(ADDRESS\_STR).Formula = FORMULA\_STR

Else

.Range(ADDRESS\_STR).Value = FORMULA\_STR

End If

i = j + 1

Loop

With Selection.Font

.Color = -4165632

.TintAndShade = 0

End With

Union(.Range("A1"), .Range("A26"), .Range("A35")).Font.Bold = True

Union(.Range("B4:B7"), .Range("B12:B17"), .Range("H22"), .Range("B33:E33"), \_

.Range("E33"), .Range("B43")).Font.Color = -4165632

End With

End Sub

'Using Eval with other UDFs

Function TFunc(A As Double, B As Double, C As Double, x As Double) As Variant

TFunc = A \* Sin(x) ^ C + B \* Cos(x) ^ C

End Function