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

'Last Update: Oct 25th, 2013

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

Option Explicit

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

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

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

'Workshop - Array Formulas in Excel

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

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

'Array formulas are a very powerful tool in Excel, allowing you to do things that are not possible with

'regular formulas. They are like ordinary formulas except that you enter them with Control/Shift/Enter

'rather than just enter. Although they may seem complicated at first, you'll find that with a little practice

'they are quite logical.

'An array formula is a formula that works with an array, or series, of data values rather than a single

'data value. There are two flavors of array formulas: first, there are those formulas that work with an

'array or series of data and aggregate it, typically using SUM, AVERAGE, or COUNT, to return a single

'value to a single cell. In this type of array formula, the result, while calculated from arrays, is a

'single value. We will examine this type of array formula first. The second flavor of array formulas is

'a formula that returns a result in to two or more cells. These types of array formulas return an array

'of values as their result.

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

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

'Section 1. Single Value Result Array Formulas

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

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

'For example, in its simple form, the formula =ROW(A1:A10) returns the number 1, which is the row number

'of the first cell in the range A1:A10. However, if this is entered as an array formula, it will return

'an array or series of numbers, each of which is the row number of a cell in the range A1:A10. That is,

'instead of returning the single value 1, it returns the array of numbers {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

'(In standard notation, arrays are written enclosed in curly braces { }.) When using array formulas, you

'typically use a container function such as SUM or COUNT to aggregate the array to a single number result.

'Expanding on the example above, the formula =SUM(ROW(A1:A10)) entered normally will return a value of 1.

'This because in its normal mode, ROW(A1:A10) returns a single number, 1, and then SUM just sums that

'single number. However, if the formula is entered as an array formula, ROW(A1:A10) returns the array of

'row numbers and then SUM adds up the elements of the array, giving a result of 55 ( = 1 + 2 + 3 + ... + 10).

'ENTERING AN ARRAY FORMULA: To enter a formula as an array formula, type the formula in the cell and press

'the CTRL SHIFT and ENTER keys at the same time rather then just ENTER. You must do this the first time

'you enter the formula and whenever you edit the formula later. If you do this properly, Excel will display

'the formula enclosed in curly braces { }. You do not type in the braces -- Excel will display them

'automatically. If you neglect to enter the formula with CTRL SHIFT ENTER, the formula may return a #VALUE

'error or return an incorrect result.

Sub Test\_Single\_Value\_Result\_Array\_Formulas()

Dim FORMULA\_STR As String

Dim DCELL As Range

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

With Range(.Cells(1, 1), .Cells(10, 1))

.Value = WorksheetFunction.Transpose(Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

End With

Set DCELL = .Cells(1, 2) 'Cell B1

With DCELL

FORMULA\_STR = "=ROW(A1:A10)"

.Formula = FORMULA\_STR

Debug.Print DCELL.Address(False, False), .Value, FORMULA\_STR

'.Clear

End With

Set DCELL = DCELL.Offset(1, 0) '.Cells(2, 2) = Cell B2

With DCELL

FORMULA\_STR = "=SUM(ROW(A1:A10))"

.Formula = FORMULA\_STR

Debug.Print DCELL.Address(False, False), .Value, FORMULA\_STR

'.Clear

End With

Set DCELL = DCELL.Offset(1, 0) '.Cells(3, 2) = Cell B3

With DCELL

FORMULA\_STR = "=SUM(ROW(A1:A10))"

.FormulaArray = FORMULA\_STR 'this is an array formula and thus must be entered with CTRL SHIFT

'ENTER in cell B3 sheet Array\_Formulas.

Debug.Print DCELL.Address(False, False), .Value, FORMULA\_STR

'.Clear

End With

DCELL.Activate

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

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

'Section 2. Formulas That Return Arrays

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

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

'The other type of array formula is one that returns an array of numbers as its result. These sort of

'array formulas are entered into multiple cells that are then treated as a group. For example, consider

'the formula =ROW(A1:A10). If this is entered into one cell, either as a normal formula or as an array

'formula, the result will be 1 in that single cell. If, however, you array enter it into a range of cells

'each cell will contain one element of the array. To do this, you first must select the range of cells

'in to which the array should be written, say C1:C10, type the formula =ROW(A1:A10), and then press CTRL

'SHIFT ENTER. The elements of the array {1, 2, ...., 10} will be written to the range of cells, with one

'element of the array in each cell. When you array enter a formula into an array of cells, Excel prevents

'you from modifying a single cell with that array range. You may select the entire range, edit the formula,

'and array-enter it again with CTRL SHIFT ENTER, but you cannot change a single element of the array.

Sub Test\_Formulas\_That\_Return\_Arrays()

Dim ADDRESS\_STR As String

Dim DCELL As Range

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

Set DCELL = Range(.Cells(1, 1), .Cells(10, 1))

ADDRESS\_STR = DCELL.Address(False, False) 'A1:A10

With DCELL

.Value = WorksheetFunction.Transpose(Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

End With

DCELL.Offset(0, 1).FormulaArray = "=ROW(A1:A10)"

DCELL.Activate

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'Some of the built-in Excel functions return an array of values. These formulas must be entered into an

'array of cells. For example, the MMULT function returns the matrix product of two arrays. The result

'is an array with the same number of rows as array1 and the same number of columns as array2.

Sub Using\_Array\_Formulas\_in\_Excel()

Dim i As Long

Dim j As Long

Dim k As Long

Dim NROW As Long

Dim NCOLUMNS As Long

Dim NSIZE As Long

Dim TEXT\_STR As String

Dim DST\_RNG As Excel.Range

k = 3: NSIZE = 10

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

Set DST\_RNG = .Cells(1, 1)

End With

'Set DST\_RNG = ActiveSheet.Cells(1, 1) 'WSHEET\_ADD\_FUNC activates the new sheet!

DST\_RNG.ColumnWidth = 40

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

'As a simple example, take the case of generating a multiplication table from a row and column of numbers,

'each from 1 to NSIZE. One way is to create a sub that enter a formula with a mixed absolute/relative address,

'and copy this over the required range, so with the numbers 1-10 (NSIZE=10) in Row 2 and Column C: =C$2\*$B3

TEXT\_STR = "Generating a Multiplication Table:"

GoSub FRAME\_CODES

For j = 1 To NSIZE

TEXT\_STR = "'" & DST\_RNG.Cells(2 + j, 3).Address(False, False) & " = " & DST\_RNG.Cells(2, 3).Address(True, False) & " \* " & \_

DST\_RNG.Cells(2 + j, 2).Address(False, True) & "; = 1 \* " & CStr(j)

DST\_RNG.Cells(2 + j, 1).Value = TEXT\_STR

For i = 1 To NSIZE

'when copied from C3 to L12 gives:

DST\_RNG.Cells(2 + i, 2 + j).Formula = "=" & DST\_RNG.Cells(2, 2 + j).Address(True, False) & "\*" & \_

DST\_RNG.Cells(2 + i, 2).Address(False, True) '

Next i

Next j

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

'The same result can be reached using an array formula: = C$15:L$15 \* $B16:$B25; = 1:10\*1:10

Set DST\_RNG = DST\_RNG.Offset(NSIZE + k)

TEXT\_STR = "Same result can be reached using an array formula:"

GoSub FRAME\_CODES

TEXT\_STR = "' = " & DST\_RNG.Cells(2 + 1, 2).Address & ":" & DST\_RNG.Cells(2 + NSIZE, 2).Address & " \* " & \_

DST\_RNG.Cells(2, 3).Address & ":" & DST\_RNG.Cells(2, 2 + NSIZE).Address & \_

"; = 1:" & CStr(NSIZE) & "\*" & "1:" & CStr(NSIZE)

DST\_RNG.Cells(2 + 1, 1).Value = TEXT\_STR 'Cell A16

'The procedure is:

'Enter the formula as shown in cell A16 (= C$15:L$15 \* $B16:$B25) and press enter as usual

'Select the entire range where output values are required: C16:L22

'Press F2 to enter edit mode

'Hold down the Ctrl and Shift keys and press Enter to enter the formula as an array formula

'The results will now display over the selected range, and the formula will show in the edit bar

'surrounded by {}

TEXT\_STR = DST\_RNG.Cells(2 + 1, 2).Address & ":" & DST\_RNG.Cells(2 + NSIZE, 2).Address & " \* " & \_

DST\_RNG.Cells(2, 3).Address & ":" & DST\_RNG.Cells(2, 2 + NSIZE).Address

Range(DST\_RNG.Cells(2 + 1, 2 + 1), DST\_RNG.Cells(2 + NSIZE, 2 + NSIZE)).FormulaArray = "=" & TEXT\_STR

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

'A similar result is given by the built-in function MMult(). Note that to get the desired result with

'MMult the order is important. Specify the row first, then the column:

Set DST\_RNG = DST\_RNG.Offset(NSIZE + k)

TEXT\_STR = "Similar result is given by the built-in function MMult():"

GoSub FRAME\_CODES

TEXT\_STR = "' = MMULT(" & DST\_RNG.Cells(2 + 1, 2).Address & ":" & DST\_RNG.Cells(2 + NSIZE, 2).Address & " \* " & \_

DST\_RNG.Cells(2, 3).Address & ":" & DST\_RNG.Cells(2, 2 + NSIZE).Address & ")" & \_

"; = MMULT(1:" & CStr(NSIZE) & "\*" & "1:" & CStr(NSIZE) & ")"

DST\_RNG.Cells(2 + 1, 1).Value = TEXT\_STR '--> Cell A29

'The procedure is:

'Enter the formula as shown in cell A29 (= MMULT($B$29:$B$38 \* $C$28:$L$28)) and press enter as usual

'Select the entire range where output values are required: C29:L38

'Press F2 to enter –edit” mode

'Hold down the Ctrl and Shift keys and press Enter to enter the formula as an array formula

'The results will now display over the selected range, and the formula will show in the edit bar surrounded

'by {}

TEXT\_STR = DST\_RNG.Cells(2 + 1, 2).Address & ":" & DST\_RNG.Cells(2 + NSIZE, 2).Address & "," & \_

DST\_RNG.Cells(2, 3).Address & ":" & DST\_RNG.Cells(2, 2 + NSIZE).Address

Range(DST\_RNG.Cells(2 + 1, 2 + 1), DST\_RNG.Cells(2 + NSIZE, 2 + NSIZE)).FormulaArray = "=MMULT(" & TEXT\_STR & ")"

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

'Important points to be aware of when using an array function are:

'Any change to the function parameters will apply to the entire output range, and must be entered with

'Ctrl-Shift-Enter (just Enter will generate an error message).

'You can extend the range of an array formula by selecting the extended range (including all of the

'original range) and pressing F2 followed by Ctrl-Shift-Enter

'You cannot reduce the range of an array formula. You must delete the entire range, then re-enter

'over the reduced range.

'You cannot modify the formula over part of the range.

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

Exit Sub

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

FRAME\_CODES:

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

With DST\_RNG

.Value = TEXT\_STR

.Font.Bold = True

End With

For j = 1 To NSIZE

With DST\_RNG

.Cells(2 + j, 2).Value = j: .Cells(2, 2 + j).Value = j ': --> New line

End With

Next j

With Union(Range(DST\_RNG.Cells(3, 2), DST\_RNG.Cells(2 + NSIZE, 2)), \_

Range(DST\_RNG.Cells(2, 3), DST\_RNG.Cells(2, 2 + NSIZE)))

.HorizontalAlignment = xlCenter

.VerticalAlignment = xlBottom

.WrapText = False

.Orientation = 0

.AddIndent = False

.IndentLevel = 0

.ShrinkToFit = False

.ReadingOrder = xlContext

.MergeCells = False

With .Interior

.Pattern = xlSolid

.PatternColorIndex = xlAutomatic

.ThemeColor = xlThemeColorDark1

.TintAndShade = -4.99893185216834E-02

.PatternTintAndShade = 0

End With

End With

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

Return

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

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

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

'Section 3. Using Index() as an array function

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

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

'This section looks at using the Excel built in Index() function to operate on arrays; it also provides a

'VBA user defined function (UDF), INDEX\_FUNC, which simplifies working with arrays within VBA, and also

'provides improved functionality when called from the spreadsheet.

'The Excel Index function returns the value of a single cell when provided with a row number and a column

'number. If either the row or the column numbers are omitted then the function will return the full

'column or row respectively. Note that the function must be entered as an array function to return all

'the data.

'As well as a single value or omitted (or 0), the Index function will accept an array to define both the

'rows or columns to be returned. The array may be a spreadsheet range (either a single row or column),

'or a list of values surrounded by {}.

'The example below shows the use of the UDF INDEX\_FUNC() and the built-in Index() function. Note that

'when using the Index() function the list of rows must be entered as a column array (either a single

'column range, or values separated by a semi-colon), and the list of columns must be entered as a row

'array (either a single row range, or values separated by a comma). If this is not done correctly the

'returned array will not be the required values.

'To simplify the input, and also to simplify the use of the function from within VBA routines, I have

'written the INDEX\_FUNC, which will accept either a row or column array (or range) to specify both rows

'or columns. Examples are shown returning the 1st, 2nd, and 4th rows and columns with a variety of

'different forms of input.

Sub Index\_as\_an\_array\_function()

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim L(1 To 3) As Integer

Dim NROWS As Integer

Dim SCOLUMN As Integer

Dim NCOLUMNS As Integer

Dim CHR\_STR As String

Dim ADDRESS0\_STR As String

Dim ADDRESS1\_STR As String

Dim ADDRESS2\_STR As String

Dim ADDRESS3\_STR As String

Dim FORMULA\_STR As String

Dim DCELL As Range

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

Set DCELL = .Cells(1, 1) 'Cell A1

With DCELL

.Value = "Using the Index function on arrays"

j = 2

NROWS = 12

For i = 1 To NROWS

.Cells(i + j, 1) = CStr(Format(DateSerial(Year(Now()), i, 1), "MMM"))

.Cells(i + j, 2) = i

.Cells(i + j, 3) = CStr(Format(DateSerial(Year(Now()), i, 1), "MMMM"))

.Cells(i + j, 4) = i + 100

Next i

ADDRESS0\_STR = Range(.Cells(1 + j, 1), .Cells(NROWS + j, 4)).Address

End With

Set DCELL = DCELL.Offset(NROWS + j + 1, 0) 'Cell A16

With DCELL

.Value = "Return a complete row or column"

j = 2

With .Cells(1 + j, 1)

.Value = "Row No:"

NCOLUMNS = 4

.Offset(0, 1).Value = NCOLUMNS

' =INDEX($A$3:$D$14,B18,)

FORMULA\_STR = "=INDEX(" & ADDRESS0\_STR & "," & .Offset(0, 1).Address(False, False) & ",)"

.Offset(1, 1).Value = "'" & FORMULA\_STR

Range(.Offset(2, 1), .Offset(2, 1 + NCOLUMNS - 1)).FormulaArray = FORMULA\_STR

End With

End With

Set DCELL = DCELL.Offset(1 + j + 1, 0) 'Cell A22

With DCELL

.Value = "Return a complete row or column"

j = 2

With .Cells(1 + j, 1)

.Value = "Column No:"

SCOLUMN = 3

.Offset(0, 1).Value = SCOLUMN

' =INDEX(A3:D14,,B22)

FORMULA\_STR = "=INDEX(" & ADDRESS0\_STR & ",," & .Offset(0, 1).Address(False, False) & ")"

.Offset(1, 1).Value = "'" & FORMULA\_STR

FORMULA\_STR = Replace(FORMULA\_STR, "$", "")

Range(.Offset(2, 1), .Offset(2 + NROWS - 1, 1)).FormulaArray = FORMULA\_STR

End With

End With

Set DCELL = DCELL.Offset(NROWS + j + 3, 0) 'Cell A37

NCOLUMNS = 3

NROWS = 3

With DCELL

.Value = "Return specified columns and/or rows"

.Offset(1, 0) = "Using INDEX\_FUNC VBA UDF"

.Offset(1, NCOLUMNS + 1) = "Using Index function"

L(1) = 1: L(2) = 2: L(3) = 4

CHR\_STR = ""

For i = 1 To NCOLUMNS - 1

CHR\_STR = CHR\_STR & L(i) & ","

Next i

CHR\_STR = CHR\_STR & L(i)

For i = 1 To 6

j = (i - 1) \* (NROWS + 2) + 2

ADDRESS1\_STR = Range(.Cells(j + 2, 1), .Cells(j + 1 + NROWS, NCOLUMNS)).Address

Debug.Print ADDRESS1\_STR

ADDRESS2\_STR = Range(.Cells(j + 2, (NCOLUMNS + 1) \* 2 + 1), \_

.Cells(j + 1 + NROWS, (NCOLUMNS + 1) \* 2 + 1)).Address

ADDRESS3\_STR = Range(.Cells(j + 2, (NCOLUMNS + 1) \* 2 + 1), \_

.Cells(j + 2, (NCOLUMNS + 1) \* 2 + NCOLUMNS)).Address

Select Case i

Case 1 '$A$40:$C$42 =INDEX($A$3:$D$14,{1,2,4},{1,2,4})

FORMULA\_STR = "=INDEX\_FUNC(" & ADDRESS0\_STR & ",{" & CHR\_STR & "},{" & CHR\_STR & "})"

Case 2 '$A$45:$C$47 =INDEX($A$3:$D$14,$I$45:$I$47,{1,2,4})

FORMULA\_STR = "=INDEX\_FUNC(" & ADDRESS0\_STR & "," & \_

Range(ADDRESS2\_STR).Address & ",{" & CHR\_STR & "})"

Range(ADDRESS2\_STR).FormulaArray = "=TRANSPOSE({" & CHR\_STR & "})"

Case 3 '$A$50:$C$52 =INDEX($A$3:$D$14,{1,2,4},$J$50:$J$52)

FORMULA\_STR = "=INDEX\_FUNC(" & ADDRESS0\_STR & ",{" & CHR\_STR & "}," & \_

Range(ADDRESS2\_STR).Offset(0, 1).Address & ")"

Range(ADDRESS2\_STR).Offset(0, 1).FormulaArray = "=TRANSPOSE({" & CHR\_STR & "})"

Case 4 '$A$55:$C$57 =INDEX($A$3:$D$14,$I$55:$I$57,$J$55:$J$57)

FORMULA\_STR = "=INDEX\_FUNC(" & ADDRESS0\_STR & "," & Range(ADDRESS2\_STR).Address & \_

"," & Range(ADDRESS2\_STR).Offset(0, 1).Address & ")"

Range(ADDRESS2\_STR).FormulaArray = "=TRANSPOSE({" & CHR\_STR & "})"

Range(ADDRESS2\_STR).Offset(0, 1).FormulaArray = "=TRANSPOSE({" & CHR\_STR & "})"

Case 5 '$A$60:$C$62 =INDEX($A$3:$D$14,$I$60:$I$62,$J$60:$L$60)

FORMULA\_STR = "=INDEX\_FUNC(" & ADDRESS0\_STR & "," & Range(ADDRESS2\_STR).Address & \_

"," & Range(ADDRESS3\_STR).Offset(0, 1).Address & ")"

Range(ADDRESS2\_STR).FormulaArray = "=TRANSPOSE({" & CHR\_STR & "})"

Range(ADDRESS3\_STR).Offset(0, 1).FormulaArray = "={" & CHR\_STR & "}"

Case 6 '$A$65:$C$67 =INDEX($A$3:$D$14,$I$65:$K$65,$L$65:$L$67)

FORMULA\_STR = "=INDEX\_FUNC(" & ADDRESS0\_STR & "," & Range(ADDRESS3\_STR).Address & \_

"," & Range(ADDRESS2\_STR).Offset(0, 3).Address & ")"

Range(ADDRESS2\_STR).Offset(0, 3).FormulaArray = "=TRANSPOSE({" & CHR\_STR & "})"

Range(ADDRESS3\_STR).FormulaArray = "={" & CHR\_STR & "}"

End Select

.Offset(j, 0) = "'" & FORMULA\_STR

Range(ADDRESS1\_STR).FormulaArray = FORMULA\_STR

.Offset(j, NCOLUMNS + 1) = "'" & Replace(FORMULA\_STR, "\_FUNC", "")

Range(ADDRESS1\_STR).Offset(0, NCOLUMNS + 1).FormulaArray = Replace(FORMULA\_STR, "\_FUNC", "")

Next i

End With

End With

Range(ADDRESS0\_STR).Activate

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'INDEX\_FUNC for slicing an array without Loop through Application.Index

'For the VBA function INDEX\_FUNC we need to slice an array i.e. fetch a row/column

'from a multidimensional array. There is no inbuilt function in VBA to do the

'same and the most common way to do so is using a loop. However it can be done

'using a worksheet function named Index.

'Syntax: Application.Index(Array, Row\_Number, Column\_Number)

'OR

'Application.WorksheetFunction.Index(Array, Row\_Number, Column\_Number)

'To extract a column from the source array, Í0' should be passed as row\_number argument.

'Similarly, to extract a row from source array, Í0' should be passed as column\_number argument.

'e.g.

'SRC\_RNG = Range(ADDRESS\_STR) e.g., ADDRESS\_STR = A1:E10

'TEMP\_ARR = Application.Index(SRC\_RNG, 2, 0)

'In the above example second row of SRC\_RNG will be fetched in TEMP\_ARR. We can also extract

'more then one row/column at the same time. In this case an array of numbers (indicating

'row/column) should be passed as row\_number / column\_number argument.

'e.g. If in the above example we need to extract 2nd, 4th and 5th rows then row number would

'be passed as an array:

'TEMP\_ARR = Application.Index(SRC\_RNG, Array(2, 4, 5), 0)

'Similarly, to extract columns:

'TEMP\_ARR = Application.Index(SRC\_RNG, 0, Application.Transpose(Array(2, 4, 5)))

'This function can also be used to fill values in a particular row/column of a range.

'Syntax: Application.Index(Range, Row\_number, Column\_number) = SourceArray / Range

'In the same example to fill the values of 2nd column of SRC\_RNG to 2nd column of range

'[A1:E10], we would do the following Application.Index([A1:E10], , 2) = Application.Index(SRC\_RNG, , 2)

Function INDEX\_FUNC(DATA\_ARR As Variant, \_

Optional NROWS As Variant = 0, \_

Optional NCOLUMNS As Variant = 0) As Variant

If TypeName(DATA\_ARR) = "Range" Then DATA\_ARR = DATA\_ARR.Value2

NROWS = GET\_VVECTOR\_FUNC(NROWS)

NCOLUMNS = GET\_HVECTOR\_FUNC(NCOLUMNS)

INDEX\_FUNC = Application.Index(DATA\_ARR, NROWS, NCOLUMNS)

End Function

Function GET\_VVECTOR\_FUNC(DATA\_ARR As Variant) As Variant

Dim TEMP\_ARR(1 To 1, 1 To 1) As Variant

If TypeName(DATA\_ARR) = "Range" Then DATA\_ARR = DATA\_ARR.Value2

If IsArray(DATA\_ARR) = False Then

TEMP\_ARR(1, 1) = DATA\_ARR

GET\_VVECTOR\_FUNC = TEMP\_ARR

Else

On Error Resume Next

If UBound(DATA\_ARR, 2) > UBound(DATA\_ARR) Then

GET\_VVECTOR\_FUNC = WorksheetFunction.Transpose(DATA\_ARR)

Else

GET\_VVECTOR\_FUNC = DATA\_ARR

End If

End If

End Function

Function GET\_HVECTOR\_FUNC(DATA\_ARR As Variant) As Variant

Dim TEMP\_ARR(1 To 1, 1 To 1) As Variant

If TypeName(DATA\_ARR) = "Range" Then DATA\_ARR = DATA\_ARR.Value2

If IsArray(DATA\_ARR) = False Then

TEMP\_ARR(1, 1) = DATA\_ARR

GET\_HVECTOR\_FUNC = TEMP\_ARR

Else

On Error GoTo NoTranspose

If UBound(DATA\_ARR, 2) < UBound(DATA\_ARR) Then

GET\_HVECTOR\_FUNC = WorksheetFunction.Transpose(DATA\_ARR)

Exit Function

End If

NoTranspose:

GET\_HVECTOR\_FUNC = DATA\_ARR

End If

End Function

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

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

'Section 4. Creating Array Formulas Using Arrays Of Data

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

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

'The IF function can be used in an array formula to test the result of multiple cell tests at one time.

'For example, you may want to compute the average of the values in A1:A5 but exclude numbers that are

'less than or equal to zero. For this, you would use an array formula with an IF function to test the

'cell values and an AVERAGE function to aggregate the result. The following formula does exactly this:

'=AVERAGE(IF(A1:A5>0,A1:A5,FALSE))

'This formula works by testing each cell in A1:A5 to > 0. This returns an array of Boolean values such

'as {TRUE, TRUE, FALSE, FALSE, TRUE}.

'A BOOLEAN VALUE is a data type that contains either the value TRUE or the value FALSE. When converted

'to numbers in an arithmetic operation, TRUE is equivalent to 1 and FALSE is equivalent to 0. Most

'arithmetic functions like SUM and AVERAGE ignore Boolean values, so those values must be converted to

'numeric values before passing them to SUM or AVERAGE.

'The IF function tests each of these results individually, and returns the corresponding value from A1:A5

'if True or the value FALSE if false. Fully expanded, the formula would look like the following:

'=AVERAGE(IF({TRUE,TRUE,FALSE,FALSE,TRUE},{A1,A2,A3,A4,A5},{FALSE,FALSE,FALSE,FALSE,FALSE})

'Note that the single FALSE value at the end of the original formula is expanded to an array of the

'appropriate size to match the array from the A1:A5 range in the formula. In array formulas, all arrays

'must be the same size. Excel will expand single elements to arrays as necessary, but will not resize

'arrays with more than one element to another size. If the arrays are not of the same size, you will

'get a #VALUE or in some cases a #N/A error.

'When the IF function evaluates, the following intermediate array is formed: {A1, A2, FALSE, FALSE, A5}.

'This is a substitution of the TRUE elements with the values from A1:A5 and the FALSE elements by FALSE.

'Since the AVERAGE function is designed within Excel to ignore Boolean values (TRUE or FALSE values),

'it will average only elements A1, A2, and A5 ignoring the TRUE and FALSE values. Note that the FALSE

'value is not converted to a zero. It is ignored completely by the AVERAGE function.

Sub Multiple\_cell\_tests\_at\_one\_time()

Dim i As Integer

Dim j() As Integer '1D Array

Dim NROWS As Integer

Dim FORMULA\_STR As String

Dim DCELL As Range

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

NROWS = 5

Set DCELL = Range(.Cells(1, 1), .Cells(NROWS, 1)) 'A1:A5

ReDim j(1 To NROWS)

j(1) = 1: j(2) = 1: j(3) = 0: j(4) = 0: j(5) = 1

For i = 1 To NROWS

DCELL.Cells(i, 1) = j(i)

Next i

Set DCELL = .Cells(1, 2) 'Cell B1

With DCELL

FORMULA\_STR = "=AVERAGE(IF(A1:A5>0,A1:A5,FALSE))"

.FormulaArray = FORMULA\_STR

Debug.Print DCELL.Address(False, False), .Value, FORMULA\_STR

End With

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'Array formulas are ideal for counting or summing cells based on multiple criteria.

'Consider the table of data shown to the right. It lists the number of products (column C) in different

'categories (column A) sold by various salesman (column B). To calculate the number of Fax machines

'sold by Brown, we can use the following array formula:

'=SUM((A2:A10="Fax")\*(B2:B10="Brown")\*(C2:C10))

'This function builds three arrays. The first array is a series of TRUE or FALSE values which are the

'results of comparing A2:A10 to the word "Fax". (Remember, Excel will expand the single "Fax" element

'to an array of items all of which are "Fax".) The second array is also a series of TRUE or FALSE values,

'the result of comparing B2:B10 to "Brown". (The single "Brown" element in the formula is expanded to an

'array of the appropriate size.) The third array is comprised of the number of units sold from the

'range C2:C10. These three arrays are multiplied together. When you multiply two arrays, the result is

'itself an array, each element of which is the product of the corresponding elements of the two arrays

'being multiplied. For example, {1, 2, 3} times {4, 5, 6} is {1\*4, 2\*5, 3\*6} = {4, 10, 18}. When TRUE

'and FALSE values are used in any arithmetic operation, they are given the values 1 and 0, respectively.

'Thus in the formula above, Excel expands the formula into the three arrays:

'(A2:A10="Fax") {TRUE, FALSE, TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE}

'(B2:B10="Brown") {TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE}

'(C2:C10) {1, 10, 20, 30, 40, 50, 60, 70, 80}

'When these array are multiplied, treating TRUE equal to 1 and FALSE equal to 0, we get the array

'{1, 0, 0, 0, 0, 0, 60, 0, 0}

'which are the quantities of Brown's two Fax sales. The SUM function simply adds up the elements of the

'array and return a result of 61, the number of Fax machines sold by Brown.

'You may have noticed that the logic of the formula tests Product equals "Fax" AND Salesman equals "Brown",

'but nowhere do we use the AND function. Here, we use multiplication to act as a logical AND function.

'Multiplication follows the same rules as the AND operator. It will return TRUE (or 1) only when both of

'the parameters are TRUE (or <> 0). If either or both parameters are FALSE (or 0), the result is FALSE

'(or 0).

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

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

'Section 5. Logical Operations With Array Formulas

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

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

'In addition to the logical AND operation using multiplication shown in the previous section, other logical

'operations can be performed arithmetically.

'A logical OR operation can be accomplished with addition. For example,

'=SUM(IF(((A2:A10="Fax")+(B2:B10="Jones"))>0,1,0))

'will count the number of sales (not the number of units sold) in which the product was a Fax OR the

'salesman was Jones (or both). Addition acts as an OR because the result it TRUE (or <> 0) if either

'one or both of the elements are TRUE (<> 0). It is FALSE ( = 0) only when both elements are FALSE

'(or 0). This formula adds two arrays: the results of the comparisons A2:A10 to "Fax", and the results

'of the comparisons B2:B10 to "Jones". Each of these arrays is an array of TRUE and FALSE values, each

'element being the result of comparing one cell to "Fax" or "Jones". It then adds these two arrays.

'When you add two arrays, the result is itself an array, each element of which is the sum of the

'corresponding element of the original arrays. For example, {1, 2, 3} + {4, 5, 6} = {1+4, 2+5, 3+6}

'= {5, 7, 9}. For each element in the sum array (A2:A10="Fax")+(B2:B10="Jones"), if that element is

'greater than 0, IF returns 1, otherwise it returns 0. Finally, SUM just adds up the array.

'An "exclusive or" or XOR operation is a comparison that returns TRUE when exactly one of the two

'elements is TRUE. XOR is FALSE if both elements are TRUE or if both elements are FALSE. Arithmetically,

'we can use the MOD operator to simulate an XOR operation. For example, to count the number of sales in

'which the product was a Fax XOR the salesman was Jones (excluding Faxes sold by Jones), we can use the

'following formula:

'=SUM(IF(MOD((A2:A10="Fax")+(B2:B10="Jones"),2),1,0))

' A "negative and" or NAND operation is a comparison that returns TRUE when neither or exactly one of

'the elements is TRUE, but returns FALSE if both elements are TRUE. For example, we can count the number

'of sales except those in which Jones sold a Fax with the formula:

'=SUM(IF((A2:A10="Fax")+(B2:B10="Jones")<>2,1,0))

Sub Counting\_summing\_cells\_based\_multiple\_criteria()

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim L As Integer

Dim NROWS As Integer

Dim NCOLUMNS As Integer

Dim DATA\_STR As String

Dim ADDRESS\_STR As String

Dim TEMP\_MATRIX As Variant

Dim FORMULA\_STR As String

Dim DCELL As Range

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

DATA\_STR = "Product,Fax,Phone,Fax,Fax,Phone,PC,Fax,Phone,PC,"

NROWS = 0

i = 1

Do

j = InStr(i, DATA\_STR, ",")

i = j + 1

NROWS = NROWS + 1

Loop Until i = 1

NROWS = NROWS - 1: NCOLUMNS = 3

ReDim TEMP\_MATRIX(1 To NROWS, 1 To NCOLUMNS)

k = 1: GoSub LOAD\_DATA

DATA\_STR = "Salesman,Brown,Smith,Jones,Smith,Jones,Smith,Brown,Davis,Jones,"

k = 2: GoSub LOAD\_DATA

DATA\_STR = "Units Sold,1,10,20,30,40,50,60,70,80,"

k = 3: GoSub LOAD\_DATA

Dim SRC\_WSHEET As Worksheet

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

Set DCELL = Range(.Cells(1, 1), .Cells(NROWS, NCOLUMNS)) 'A1:C10

DCELL.Value = TEMP\_MATRIX

ADDRESS\_STR = DCELL.Address(False, False)

End With

'A2:A10 = Range(Range(ADDRESS\_STR).Cells(2, 1), Range(ADDRESS\_STR).Cells(NROWS, 1)).Address(False, False)

'B2:C10 = Range(Range(ADDRESS\_STR).Cells(2, 2), Range(ADDRESS\_STR).Cells(NROWS, 2)).Address(False, False)

'C2:C10 = Range(Range(ADDRESS\_STR).Cells(2, 3), Range(ADDRESS\_STR).Cells(NROWS, 3)).Address(False, False)

Set DCELL = DCELL.Cells(1, 1).Offset(NROWS + 1, 1)

With DCELL

'SUM((A2:A10="Fax")\*(B2:B10="Brown")\*(C2:C10))

FORMULA\_STR = "=SUM((" & \_

Range(Range(ADDRESS\_STR).Cells(2, 1), \_

Range(ADDRESS\_STR).Cells(NROWS, 1)).Address(False, False) & \_

"=" & """" & "Fax" & """" & ")\*(" & \_

Range(Range(ADDRESS\_STR).Cells(2, 2), \_

Range(ADDRESS\_STR).Cells(NROWS, 2)).Address(False, False) & \_

"=" & """" & "Brown" & """" & ")\*(" & \_

Range(Range(ADDRESS\_STR).Cells(2, 3), \_

Range(ADDRESS\_STR).Cells(NROWS, 3)).Address(False, False) & \_

"))"

i = 1

.Cells(i + 0, 1).Value = "Summing Sales: Faxes Sold By Brown"

.Cells(i + 1, 2).FormulaArray = FORMULA\_STR

.Cells(i + 1, 4).Value = "'" & FORMULA\_STR '.Cells(i + 1, 2).Formula

'SUM((A2:A10="Fax")\*(B2:B10="Brown"))

FORMULA\_STR = "=SUM((" & \_

Range(Range(ADDRESS\_STR).Cells(2, 1), \_

Range(ADDRESS\_STR).Cells(NROWS, 1)).Address(False, False) & \_

"=" & """" & "Fax" & """" & ")\*(" & \_

Range(Range(ADDRESS\_STR).Cells(2, 2), \_

Range(ADDRESS\_STR).Cells(NROWS, 2)).Address(False, False) & \_

"=" & """" & "Brown" & """" & "))"

i = i + 3

.Cells(i + 0, 1).Value = "Logical AND (Faxes And Jones)"

.Cells(i + 1, 2).FormulaArray = FORMULA\_STR

.Cells(i + 1, 4).Value = "'" & FORMULA\_STR '.Cells(i + 1, 2).Formula

'SUM(IF((A2:A10="Fax")+(B2:B10="Jones"),1,0))

FORMULA\_STR = "=SUM(IF((" & \_

Range(Range(ADDRESS\_STR).Cells(2, 1), \_

Range(ADDRESS\_STR).Cells(NROWS, 1)).Address(False, False) & \_

"=" & """" & "Fax" & """" & ")+(" & \_

Range(Range(ADDRESS\_STR).Cells(2, 2), \_

Range(ADDRESS\_STR).Cells(NROWS, 2)).Address(False, False) & \_

"=" & """" & "Jones" & """" & "),1,0))"

i = i + 3

.Cells(i + 0, 1).Value = "Logical OR (Faxes Or Jones)"

.Cells(i + 1, 2).FormulaArray = FORMULA\_STR

.Cells(i + 1, 4).Value = "'" & FORMULA\_STR '.Cells(i + 1, 2).Formula

'SUM(IF(MOD((A2:A10="Fax")+(B2:B10="Jones"),2),1,0))

FORMULA\_STR = "=SUM(IF(MOD((" & \_

Range(Range(ADDRESS\_STR).Cells(2, 1), \_

Range(ADDRESS\_STR).Cells(NROWS, 1)).Address(False, False) & \_

"=" & """" & "Fax" & """" & ")+(" & \_

Range(Range(ADDRESS\_STR).Cells(2, 2), \_

Range(ADDRESS\_STR).Cells(NROWS, 2)).Address(False, False) & \_

"=" & """" & "Jones" & """" & "),2),1,0))"

i = i + 3

.Cells(i + 0, 1).Value = "Logical XOR (Fax Or Jones but not both)"

.Cells(i + 1, 2).FormulaArray = FORMULA\_STR

.Cells(i + 1, 4).Value = "'" & FORMULA\_STR '.Cells(i + 1, 2).Formula

'SUM(IF((A2:A10="Fax")+(B2:B10="Jones")<>2,1,0))

FORMULA\_STR = "=SUM(IF((" & \_

Range(Range(ADDRESS\_STR).Cells(2, 1), \_

Range(ADDRESS\_STR).Cells(NROWS, 1)).Address(False, False) & \_

"=" & """" & "Fax" & """" & ")+(" & \_

Range(Range(ADDRESS\_STR).Cells(2, 2), \_

Range(ADDRESS\_STR).Cells(NROWS, 2)).Address(False, False) & \_

"=" & """" & "Jones" & """" & ")<>2,1,0))"

i = i + 3

.Cells(i + 0, 1).Value = "Logical NAND (All Sales Except Fax And Jones)"

.Cells(i + 1, 2).FormulaArray = FORMULA\_STR

.Cells(i + 1, 4).Value = "'" & FORMULA\_STR '.Cells(i + 1, 2).Formula

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

Exit Sub

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

LOAD\_DATA:

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

i = 1

For L = 1 To NROWS

j = InStr(i, DATA\_STR, ",")

TEMP\_MATRIX(L, k) = Mid(DATA\_STR, i, j - i)

i = j + 1

Next L

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

Return

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

End Sub

'Other Useful Array Functions: Array formulas can do a wide variety of tasks. A few miscellaneous array

'formulas are shown below:

'Sum Ignoring Errors

'Normally, if there is an error in a cell, the SUM function will return that error. The following formula

'will ignore the error values.

'=SUM(IF(ISERROR(A1:A10),0,A1:A10))

'Average Ignoring Errors

'This formula will ignore errors when averaging range.

'=AVERAGE(IF(ISERROR(A1:A10),FALSE,IF(A1:A10="",FALSE,A1:A10)))

'Average Ignoring Zeros

'This formula will ignore zero values in an AVERAGE function.

'=AVERAGE(IF(A1:A10<>0,A1:A10,FALSE))

'Sum Of Absolute Values

'You can sum a range of number treating them all as positive using the ABS function.

'=SUM(ABS(A1:A10))

'Sum Of Integer Portion Only

'This formula will sum only the integer portion of the numbers in A1:A10. The fractional portion is discarded.

'=SUM(TRUNC(A1:A5))

'Longest Text In Cells

'This formula will return the contents of the cell with the longest amount of text in it.

'=OFFSET(A1,MATCH(MAX(LEN(A1:A10)),LEN(A1:A10),0)-1,0,1,1)

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

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

'Section 6. Creating Sequences And Loops For Array Formulas

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

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

'When you are constructing some types of array formulas, you need to create a sequence of numbers for a

'function to process as an array. As an example, consider an array formula that will compute the average

'of the Nth largest elements in a range. To do this, we will use the LARGE function to get the largest

'numbers, and then pass those numbers as an array to AVERAGE to compute the average. Normally, the LARGE

'function takes as parameters a range to process and a number indicating which largest value to return

'(1 = largest, 2 = second largest, etc.,). But LARGE does work with arrays for its second parameter. You

'might be tempted to type in the array in the formula yourself: =LARGE(A1:A10,{1,2,3}). While this will

'indeed work, it is tedious.

'Instead, you can use the ROW function to return a sequence of numbers. When used in an array formula, the

'function ROW(m:n) will return an array of integers from m to n. Therefore, we can use ROW to create the

'array to pass to LARGE. This changes our array formula to =LARGE(A1:A10,ROW(1:3)). This brings us closer

'to a good formula, but two things remain.

'First, if you insert a row between rows 1 through 3, Excel will change the row reference 1:3, and

'therefore the formula will average the wrong numbers. Second, the formula is locked into the three

'largest values. We can make it more flexible by making the number of elements to average a cell reference

'that can be easily changed. For example, we can specify that cell C1 contains the size of the array to

'pass to LARGE. This is accomplished with the INDIRECT function. (Click here for more information about

'INDIRECT.) The INDIRECT function converts a string representing a cell reference into an actual cell

'reference. The sub-formula ROW(INDIRECT("1:"&C1)) will return an array of numbers between 1 and the

'value in cell C1. Now, coming together the formula to average the N largest values in A1:A10 becomes:

Sub Creating\_sequences\_and\_loops\_for\_array\_formulas()

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim NROWS As Integer

Dim ADDRESS\_STR As String

Dim FORMULA\_STR As String

Dim DCELL As Range

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

NROWS = 11

k = NROWS / 2 '5 largest values

Set DCELL = Range(.Cells(1, 1), .Cells(NROWS, 1))

ADDRESS\_STR = DCELL.Address(False, False) 'A1:A10

j = 5

For i = 1 To NROWS

DCELL.Cells(i, 1).Value = Math.Rnd

DCELL.Cells(i, j).Value = DCELL.Cells(i, 1).Value

Next i

'Sort Range

DCELL.Offset(0, j - 1).Sort Key1:=DCELL.Offset(0, j - 1).Cells(1, 1), Order1:=xlAscending, Header:=xlNo

Set DCELL = .Cells(1, 2) 'Cell B1

With DCELL

.Cells(1, 2) = k

FORMULA\_STR = "=AVERAGE(LARGE(" & ADDRESS\_STR & ",ROW(INDIRECT(" & """" & \_

"1:" & """" & "&" & .Cells(1, 2).Address(False, False) & "))))"

.FormulaArray = FORMULA\_STR 'AVERAGE(LARGE(A1:A10,ROW(INDIRECT("1:"&C1))))

Debug.Print DCELL.Address(False, False), .Value, FORMULA\_STR

With .Cells(1, j)

.Formula = "=AVERAGE(" & Range( \_

Range(ADDRESS\_STR).Offset(0, j - 1).Cells(NROWS - k + 1, 1), \_

Range(ADDRESS\_STR).Offset(0, j - 1).Cells(NROWS, 1)).Address(False, False) & ")"

.Offset(0, 1).Formula = "=" & .Address(False, False) & "=" & \_

Range(ADDRESS\_STR).Cells(1, 2).Address(False, False)

End With

End With

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

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

'Section 7. Count Distinct, Unique And Successive Values Using FREQUENCY()

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

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

'In my opinion, one of the hardest worksheet functions to get the hang of is the FREQUENCY() function. For

'this section, IÍve assembled a collection of FREQUENCY() formulae which give a count of distinct, unique

'or successive values. I use ñdistinctî to mean one copy of each different number (or string), and ñuniqueî

'to mean numbers or strings that occur only once. So in the list ñ1,1,1,2,2,3? there are 3 distinct

'numbers (1,2,3) but only one unique number (3).

'In this section we will count the number of occurrences of different numbers or strings, rather than

'return a list of them (See section 8.), using the Frequency() function is the way to do it.

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

'7.1 Syntax:

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

'The syntax of the FREQUENCY() function is given below for reference:

'FREQUENCY(data\_array, bins\_array)

'i) Data\_array Required. An array of or reference to a set of values for which you want to count frequencies.

'If data\_array contains no values, FREQUENCY() returns an array of zeros.

'ii) Bins\_array Required. An array of or reference to intervals into which you want to group the values in

'data\_array. If bins\_array contains no values, FREQUENCY() returns the number of elements in data\_array.

'iii) FREQUENCY() is entered as an array formula after you select a range of adjacent cells into which you want

'the returned distribution to appear.

'iv) The number of elements in the returned array is one more than the number of elements in bins\_array. The

'extra element in the returned array returns the count of any values above the highest interval. For

'example, when counting three ranges of values (intervals) that are entered into three cells, be sure

'to enter FREQUENCY() into four cells for the results. The extra cell returns the number of values in

'data\_array that are greater than the third interval value.

'v) FREQUENCY() ignores blank cells and text.

'vi) Formulas that return arrays must be entered as array formulas.

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

'7.2 Count Of Distinct Values

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

'The formulae weÍre going to explore in this section form the backbone of the formulae used throughout the rest

'of the section, so IÍm going to explain them by breaking them into bite-size chunks.

'7.2.1 Numbers only

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

'This is one of the simplest formulae weÍll be looking at today.

'We have a list of numbers in B4:B12 and we want to determine how many

'distinct values there are.

Sub Test\_Distinct\_Numbers\_Only()

Dim DST\_RNG As Range

Dim FORMULA\_STR As String

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, 3, 1, 7, 2, 2, 1, 1)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=SUMPRODUCT(--(FREQUENCY(" & DST\_RNG.Address & "," & DST\_RNG.Address & ")>0))"

Debug.Print FORMULA\_STR

.Cells(4, 5).Formula = FORMULA\_STR

'E4 = Count of distinct values

'The distinct numbers in the list are 1, 2, 3 and 7, so the count of distinct numbers

'is 4. LetÍs break the formula down step-by-step to see how it works.

'=SUMPRODUCT(--(FREQUENCY(B4:B12,B4:B12)>0))

'Firstly, letÍs substitute the cell references for the cell values. Note that SUMPRODUCT()

'sets the context of the formula so that Excel understands it has to be processed like

'an array formula, meaning that this one doesnÍt have to be CTRL+SHIFT+ENTERÍd.

'=SUMPRODUCT(--(FREQUENCY({1;2;3;1;7;2;2;1;1},{1;2;3;1;7;2;2;1;1})>0))

'Next letÍs evaluate the FREQUENCY() part of the formula. The key points to this are:

'-> Duplicated bins return 0.

'-> There are nine bins but the FREQUENCY() function returns ten values

'in the array. The last bucket is a count of any values which are greater than the highest

'specified interval.

'=SUMPRODUCT(--({4;3;1;0;1;0;0;0;0;0}>0))

'So the four at the start of the array tells us there are four 1s in the list. The three tells

'us there are three 2s in the list, and so on.

'Now, weÍre not interested how many 1s, 2s etc there are in the list, just simply that thereÍs

'at least one of each of them. We compare every element in the array using >0 to give us an

'array of logical values.

'=SUMPRODUCT(--{TRUE;TRUE;TRUE;FALSE;TRUE;FALSE;FALSE;FALSE;FALSE;FALSE})

'Next, the two unary minus operators coerce the TRUE values to 1s and the FALSE values to 0s.

'We have to do this because SUMPRODUCT() canÍt sum logical values but it can sum numbers.

'=SUMPRODUCT({1;1;1;0;1;0;0;0;0;0})

'Then finally the array is summed to give an answer of 4.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'7.2.2 Text and/or Numbers

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

'If you want to get a distinct count where text values are to be included then life becomes a little more

'complicated because FREQUENCY() ignores text values.

Sub Test\_Distinct\_Text\_and\_or\_Numbers()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, "aa", "aa", "e", "", "b", 3, 3)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=SUM(--(FREQUENCY(IF(" & \_

DST\_RNG.Address & "<>" & """" & """" & ",MATCH(" & \_

"""" & "~" & """" & "&" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & "&" & """" & """" & ",0)),ROW(" & \_

DST\_RNG.Address & ")-ROW(" & DST\_RNG.Cells(1).Address(False, False) & ")+1)>0))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR

'E4 = Count of distinct values

'HereÍs the formula from the formula bar:

'=SUM(--(FREQUENCY(IF(B4:B12<>"",MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1)>0))

'This formula will work fine if your data just contains numbers too, so you might be wondering

'why I suggested a different formula for the numbers-only scenario in section 2.1: the answer

'is that this formula is significantly slower (over larger ranges) than the one in 2.1, so use

'that one when youÍve only got numbers. This theme repeats throughout the section.

'LetÍs break the formula down:

'Because there is an IF() function inside the FREQUENCY() formula, we have to use a SUM() array

'formula rather than SUMPRODUCT(). To enter an array formula, type it into the formula bar and

'then press CTRL+SHIFT+ENTER. If done correctly, Excel will automatically surround the formula

'with { } parentheses. You cannot enter the parentheses manually. You can see the parentheses in

'the formula bar in cell E4.

'The B4:B12<>"" check is used so that blank cells are ignored. It returns this array:

'{TRUE;TRUE;TRUE;TRUE;TRUE;FALSE;TRUE;TRUE;TRUE}.

'The tilde ~ in the MATCH() function is used as a precaution against a wildcard being the starting

'character in one of the cells. If youÍre sure that your values wonÍt ever start with these

'problematic characters (which are \* ? and ~) then you can replace "~"& with just ""&. In turn,

'if youÍre certain that your cell values are all text data types then you donÍt need the ""&

'either so you can simply remove the "~"& from the formula altogether. ItÍs also worth noting that

'the formula can fail if any of the cells contain a wildcard (particularly \*) elsewhere within the text.

'The values in the lookup\_array parameter of the MATCH() function are coerced to text data types by

'using &"". If your cell values are only text data types then you can remove it.

'So the MATCH() function returns the position of the first occurrence of each value: {1;2;3;3;5;6;7;8;8}.

'Taking the above into account, the IF() part of the formula evaluates to:

'IF({TRUE;TRUE;TRUE;TRUE;TRUE;FALSE;TRUE;TRUE;TRUE},{1;2;3;3;5;6;7;8;8})

'Which, in turn, evaluates to:

'{1;2;3;3;5;FALSE;7;8;8}

'The FALSE value corresponds to the blank cell.

'Next, the ROW(B4:B12)-ROW(B4)+1 part of the formula returns an array of sequential integers like

'{1;2;3;4;5;6;7;8;9}, so we can evaluate the formula to:

'=SUM(--(FREQUENCY({1;2;3;3;5;FALSE;7;8;8},{1;2;3;4;5;6;7;8;9})>0))

'From there, itÍs just a case of following the same steps as the previous formula to arrive at a

'result of 6:

'=SUM(--({1;1;2;0;1;0;1;2;0;0}>0))

'evaluates to:

'=SUM(--{TRUE;TRUE;TRUE;FALSE;TRUE;FALSE;TRUE;TRUE;FALSE;FALSE})

'evaluates to:

'=SUM({1;1;1;0;1;0;1;1;0;0})

'evaluates to:

'=6

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.3 Count Of Unique Values

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

'ïUniqueÍ values are different to ïdistinctÍ values because they are values which only occur once.

'7.3.1 Numbers only

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

'In the example below the unique numbers are 3 and 7.

Sub Test\_Unique\_Numbers\_Only()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, 3, 1, 7, 2, 2, 1, 1)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=SUMPRODUCT(--(FREQUENCY(" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & ")=1))"

Debug.Print FORMULA\_STR

.Cells(4, 5).Formula = FORMULA\_STR 'E4 = Count of unique values

'=SUMPRODUCT(--(FREQUENCY(B4:B12,B4:B12)=1))

'This formula is exactly the same as the formula in section 4.7.2.1 except that the >0 has been changed to =1.

'This is because we only want a count of values which occur exactly once.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'7.3.2 Text and/or Numbers

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

'In the example below, the unique values are 1, 2, ñeî and ñbî.

Sub Test\_Unique\_Text\_and\_or\_Numbers()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, "aa", "aa", "e", "", "b", 3, 3)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=SUM(--(FREQUENCY(IF(" & \_

DST\_RNG.Address & "<>" & """" & """" & ",MATCH(" & \_

"""" & "~" & """" & "&" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & "&" & """" & """" & ",0)),ROW(" & \_

DST\_RNG.Address & ")-ROW(" & DST\_RNG.Cells(1).Address(False, False) & ")+1)=1))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count of unique values

'=SUM(--(FREQUENCY(IF(B4:B12<>"",MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1)=1))

'This formula is exactly the same as the formula in section 4.7.2.2 except that the >0 has been changed to =1.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.4 Count Of Distinct Values With Condition

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

'Now we are going to add an extra condition into our formulae. In the example below, the distinct values

'corresponding to the letter ñbî are 2, ñaaî and 3, so we want to calculate a distinct count of 3.

Sub Test\_Distinct\_Values\_With\_Condition()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, "aa", "aa", "e", "", "b", 3, 3)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

DATA\_ARR = Array("a", "b", "a", "b", "a", "b", "a", "b", "a")

DST\_RNG.Offset(0, -1).Value = WorksheetFunction.Transpose(DATA\_ARR)

.Cells(4, 4).Value = "b" 'd4 condition

FORMULA\_STR = "=SUM(--(FREQUENCY(IF((" & \_

DST\_RNG.Offset(0, -1).Address & "=" & .Cells(4, 4).Address & ")\*(" & \_

DST\_RNG.Address & "<>" & """" & """" & "),MATCH(" & \_

"""" & "~" & """" & "&" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & "&" & """" & """" & ",0)),ROW(" & \_

DST\_RNG.Address & ")-ROW(" & DST\_RNG.Cells(1).Address(False, False) & ")+1)>0))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count of distinct values

'=SUM(--(FREQUENCY(IF((A4:A12=D4)\*(B4:B12<>""), MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1)>0))

'This formula is very similar to the formula in 4.7.3.2. The only difference is that an extra ïandÍ

'condition has been specified by including (A4:A12=D4)\* in the formula. You can add more ïandÍ conditions

'if you like by including additional (condition)\*.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.5 Count Of Unique Values With Condition

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

'Here we have exactly the same values as before, except we want to count only the unique values corresponding

'to ñbî. In this case, 2, ñaaî and 3 only occur once each when theyÍre on the same row as the letter ñbî,

'so we want to calculate a unique count of 3.

Sub Test\_Unique\_Values\_With\_Condition()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, "aa", "aa", "e", "", "b", 3, 3)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

DATA\_ARR = Array("a", "b", "a", "b", "a", "b", "a", "b", "a")

DST\_RNG.Offset(0, -1).Value = WorksheetFunction.Transpose(DATA\_ARR)

.Cells(4, 4).Value = "b" 'd4 condition

FORMULA\_STR = "=SUM(--(FREQUENCY(IF((" & \_

DST\_RNG.Offset(0, -1).Address & "=" & .Cells(4, 4).Address & ")\*(" & \_

DST\_RNG.Address & "<>" & """" & """" & "),MATCH(" & \_

"""" & "~" & """" & "&" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & "&" & """" & """" & ",0)),ROW(" & \_

DST\_RNG.Address & ")-ROW(" & DST\_RNG.Cells(1).Address(False, False) & ")+1)=1))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count of distinct values

'=SUM(--(FREQUENCY(IF((A4:A12=D4)\*(B4:B12<>""), MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1)=1))

'This formula is exactly the same as the formula in section 4.7.4 except that the >0 has been changed to =1.

'This is because we only want a count of values which occur exactly once.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.6 Count Of Maximum Occurrences

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

'If we want to get the number of occurrences of a given value from a list then we can use COUNTIF(). But what

'if we want to get the count of whichever value which occurs the most?

'7.6.1 Numbers only

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

'In the example below, the number 1 occurs on four occasions, which is more than any other number.

Sub Test\_Count\_Maximum\_Occurrences\_Numbers\_Only()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, 3, 1, 7, 2, 2, 1, 1)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=MAX(FREQUENCY(" & DST\_RNG.Address & "," & DST\_RNG.Address & "))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count\_Maximum\_Occurrences\_Numbers\_Only

'=MAX(FREQUENCY(B4:B12,B4:B12))

'FREQUENCY(B4:B12,B4:B12) returns {4;3;1;0;1;0;0;0;0;0} and the MAX() function takes the largest value

'from the array.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'7.6.2 Text and/or Numbers

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

'In the example below, the values which occur the most are îaaî and 3, with each occurring twice.

Sub Test\_Count\_Maximum\_Occurrences\_Text\_and\_or\_Numbers()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, "aa", "aa", "e", "", "b", 3, 3)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=MAX(FREQUENCY(IF(" & \_

DST\_RNG.Address & "<>" & """" & """" & ",MATCH(" & \_

"""" & "~" & """" & "&" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & "&" & """" & """" & ",0)),ROW(" & \_

DST\_RNG.Address & ")-ROW(" & DST\_RNG.Cells(1).Address(False, False) & ")+1))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count\_Maximum\_Occurrences\_Text\_and\_or\_Numbers

End With

'=MAX(FREQUENCY(IF(B4:B12<>"",MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1))

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.7 Count Of Minimum Occurrences

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

'Getting the minimum number of occurrences isnÍt as straightforward as getting the maximum because the

'FREQUENCY() function returns 0s in the array which we effectively want to ignore because theyÍre for the

'duplicate buckets.

'7.7.1 Numbers only

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

'Here the numbers 3 and 7 occur only once, so the minimum number of occurrences should be 1.

Sub Test\_Count\_Minimum\_Occurrences\_Numbers\_Only()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, 3, 1, 7, 2, 2, 1, 1)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=MIN(IF(FREQUENCY(" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & ")>0,FREQUENCY(" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & ")))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count\_Minimum\_Occurrences\_Numbers\_Only

'=MIN(IF(FREQUENCY(B4:B12,B4:B12)>0,FREQUENCY(B4:B12,B4:B12)))

'We work around the problem of the duplicate buckets by checking the values returned by FREQUENCY() and only

'considering those which are greater than zero. The FREQUENCY() function returns {4;3;1;0;1;0;0;0;0;0} so

'the formula evaluates to:

'=MIN(IF({4;3;1;0;1;0;0;0;0;0}>0,{4;3;1;0;1;0;0;0;0;0}))

'Which, in turn, evaluates to:

'=MIN({4;3;1;FALSE;1;FALSE;FALSE;FALSE;FALSE;FALSE})

'MIN() ignores logical values so it picks up 1 from the array.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'7.7.2 Text and/or Numbers

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

'The values 1, 2, ñeî and ñbî occur only one, so the minimum number of occurrences is 1.

Sub Test\_Count\_Minimum\_Occurrences\_Text\_and\_or\_Numbers()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, "aa", "aa", "e", "", "b", 3, 3)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

FORMULA\_STR = "=MIN(IF(FREQUENCY(IF(" & DST\_RNG.Address & "<>" & \_

"""" & """" & ",MATCH(" & """" & "~" & """" & "&" & DST\_RNG.Address & "," & \_

DST\_RNG.Address & "&" & """" & """" & ",0)),ROW(" & DST\_RNG.Address & \_

")-ROW(" & DST\_RNG.Cells(1).Address(False, False) & \_

")+1),FREQUENCY(IF(" & DST\_RNG.Address & "<>" & """" & """" & \_

",MATCH(" & """" & "~" & """" & "&" & DST\_RNG.Address & "," & DST\_RNG.Address & \_

"&" & """" & """" & ",0)),ROW(" & DST\_RNG.Address & ")-ROW(" & \_

DST\_RNG.Cells(1).Address(False, False) & ")+1)))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Count\_Minimum\_Occurrences\_Text\_and\_or\_Numbers

'The formula in the formula bar is:

'=MIN(IF(FREQUENCY(IF(B4:B12<>"",MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1),

'FREQUENCY(IF(B4:B12<>"",MATCH("~"&B4:B12,B4:B12&"",0)),ROW(B4:B12)-ROW(B4)+1)))

'This evaluates to:

'=MIN(IF(FREQUENCY({1;2;3;3;5;FALSE;7;8;8},{1;2;3;4;5;6;7;8;9}),

'FREQUENCY({1;2;3;3;5;FALSE;7;8;8},{1;2;3;4;5;6;7;8;9})))

'then to:

'=MIN(IF({1;1;2;0;1;0;1;2;0;0},{1;1;2;0;1;0;1;2;0;0}))

'then to:

'=MIN({1;1;2;FALSE;1;FALSE;1;2;FALSE;FALSE})

'finally arriving at:

'=1

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.8 Maximum Successions Of A Given Value

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

'Given the list of numbers below, we want to calculate the largest number of times where the number 1

'repeats in succession.

Sub Test\_Maximum\_Successions\_Of\_A\_Given\_Value()

Dim FORMULA\_STR As String

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

DATA\_ARR = Array(1, 2, 3, 1, 7, 2, 2, 1, 1)

Set DST\_RNG = Range(.Cells(4, 2), .Cells(12, 2)) 'B4:B12

DST\_RNG.Value = WorksheetFunction.Transpose(DATA\_ARR)

.Cells(4, 4) = 1

FORMULA\_STR = "=MAX(FREQUENCY(IF(" & DST\_RNG.Address & "=" & \_

.Cells(4, 4).Address & ",ROW(" & DST\_RNG.Address & ")),IF(" & \_

DST\_RNG.Address & "<>" & .Cells(4, 4).Address & ",ROW(" & \_

DST\_RNG.Address & "))))"

Debug.Print FORMULA\_STR

.Cells(4, 5).FormulaArray = FORMULA\_STR 'E4 = Max successive values

'=MAX(FREQUENCY(IF($B$4:$B$12=D4,ROW($B$4:$B$12)),IF($B$4:$B$12<>D4,ROW($B$4:$B$12))))

'If we evaluate the IF() functions nested inside the FREQUENCY() function we get:

'=MAX(FREQUENCY({4;FALSE;FALSE;7;FALSE;FALSE;FALSE;11;12},{FALSE;5;6;FALSE;8;9;10;FALSE;FALSE}))

'This then evaluates to:

'=MAX({1;0;1;0;0;2})

'The 2 at the end of the array is the largest consecutive count of 1s.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

'7.9 Example for non-contiguous ranges

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

'As you can see, once you have to start dealing with text values, the FREQUENCY() formulae become much more

'complicated and slower. When formulae using native functions become long, complicated and slow, they also

'become prime candidates for conversion into VBA UDFs. With that in mind, in the next sections weÍll have a

'go at writing a couple of VBA UDFs to count distinct and unique values. In the meantime, here it is one

'example for working with non-contiguous ranges.

'Imagine you have some phone call information about three people working in a customer services team.

'The count of phone calls is split into two types, ABC and DEF, for each person and you need to get the

'ABC mean average from the data set. But the average should not take into account any 0 values in the cells

'so the correct answer would be 56: (24 + 21 + 99 + 67 + 87 + 6 + 88) / 7 = 56

'In this case you may want to do a conditional average: you want to ignore cells which contain 0's. Usually you

'could do this by using the AVERAGEIF() function but you canÍt do that here because the ABC data is in

'three separate, or non-contiguous, ranges. This formula will return a #VALUE! error because AVERAGEIF()

'canÍt handle non-contiguous ranges: =AVERAGEIF((B3:B7,D3:D7,F3:F7),"<>0")

'To get an average from a non-contiguous set of ranges you could usually use the SUM() and COUNT() worksheet

'functions like so: =SUM(B3:B7,D3:D7,F3:F7)/COUNT(B3:B7,D3:D7,F3:F7). But the problem is the COUNT() will include

'the 0's, so the formula returns a result of 26.13 which is incorrect. You could try to adjust the formula to

'ignore zeros by using COUNTIF() instead of COUNT(), but COUNTIF() canÍt be used on non-contiguous ranges

'either so this will also return #VALUE!: =SUM(B3:B7,D3:D7,F3:F7)/COUNTIF((B3:B7,D3:D7,F3:F7),"<>0")

'So the question is how can you get a count of non-zero values from those cells?

'The solution to this quandary is as follows:

Sub Test\_Phone\_Call\_Information()

Dim ABC\_ARR As Variant

Dim DEF\_ARR As Variant

Dim DST\_RNG As Range

Dim DATA\_ARR As Variant

Dim FORMULA\_STR As String

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

FORMULA\_STR = "=SUM(B3:B7,D3:D7,F3:F7)/INDEX(FREQUENCY((B3:B7,D3:D7,F3:F7),0),2)"

Debug.Print FORMULA\_STR

'The SUM() part of the formula is straightforward enough, but letÍs have a closer look at the rest of it. If

'you feel that you need a quick refresher on the FREQUENCY() function then take a look at this section.

'FREQUENCY((B3:B7,D3:D7,F3:F7),0)

'The key point is that FREQUENCY() can be used on non-contiguous ranges: the data\_array is (B3:B7,D3:D7,F3:F7).

'The bins\_array is 0, so FREQUENCY() will return an array of two values: a count of the ABC numbers which fall

'into the first bucket of 0, and a count of the ABC numbers which are greater than 0:

'FREQUENCY((B3:B7,D3:D7,F3:F7),0)

'evaluates to

'{8;7}

'ItÍs a safe assumption that there will not be any negative numbers in this particular data set so the 8 is the

'count of values which are exactly equal to 0 and the 7 is the count of values which are greater than 0. The 7

'is the number weÍre interested in, so next we use the INDEX() function to extract it from the array:

'INDEX({8;7},2)

'evaluates to

'{7}

'and then you divide the sum of the values by 7 to get the correct answer of 56.

'SUM(B3:B7,D3:D7,F3:F7)/{7}

'evaluates to:

'392/{7}

'evaluates to your final answer:

'{56}

'The formula will still work if there are empty cells or even if the non-contiguous ranges are different

'sizes.

ABC\_ARR = Array("Date", Now, Now + 1, Now + 2, Now + 3, Now + 4)

Range(.Cells(2, 1), .Cells(7, 1)).Value = WorksheetFunction.Transpose(ABC\_ARR)

.Cells(1, 2) = "Charles"

ABC\_ARR = Array("ABC", 24, 21, 0, 99, 0)

Range(.Cells(2, 2), .Cells(7, 2)).Value = WorksheetFunction.Transpose(ABC\_ARR)

DEF\_ARR = Array("DEF", 48, 9, 0, 0, 0)

Range(.Cells(2, 3), .Cells(7, 3)).Value = WorksheetFunction.Transpose(DEF\_ARR)

.Cells(1, 4) = "Mike"

ABC\_ARR = Array("ABC", 67, 0, 0, 0, 0)

Range(.Cells(2, 4), .Cells(7, 4)).Value = WorksheetFunction.Transpose(ABC\_ARR)

DEF\_ARR = Array("DEF", 1, 1, 60, 28, 1)

Range(.Cells(2, 5), .Cells(7, 5)).Value = WorksheetFunction.Transpose(DEF\_ARR)

.Cells(1, 6) = "Kate"

ABC\_ARR = Array("ABC", 87, 0, 6, 88, 0)

Range(.Cells(2, 6), .Cells(7, 6)).Value = WorksheetFunction.Transpose(ABC\_ARR)

DEF\_ARR = Array("DEF", 45, 40, 88, 14, 0)

Range(.Cells(2, 7), .Cells(7, 7)).Value = WorksheetFunction.Transpose(DEF\_ARR)

.Cells(1, 1).Formula = FORMULA\_STR 'Note that your suggested formula does not need to be CTRL+SHIFT+ENTERÍd.

If (.Cells(1, 1).Value - WorksheetFunction.Sum(Range("B3:B7"), Range("D3:D7"), Range("F3:F7")) / 7) > 0 Then

Debug.Print "Try Again!"

Else

Debug.Print "Correct Formulation"

End If

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

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

'Section 8. How to return multiple values with Array Formulas

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

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

'The VLOOKUP function in Excel is designed to return only the corresponding value of the first instance of

'a lookup value. But there is a work-around to identify multiple matches. Although VLOOKUP is not used in

'these examples, these array formulas are easier to understand and troubleshoot.

'-> Excel Functions in this section:

'1) VLOOKUP(lookup\_value; table\_array; col\_index\_num; [range\_lookup])

'Looks for a value in the leftmost column of a table, and then returns a value in the same row from a column you

'specify. By default, the table must be sorted in ascending order.

'2) IF(logical\_test;[value\_if:true];[value\_if\_false])

'Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE

'3) SMALL(array,k)

'Returns the k-th smallest number in this data set.

'4) INDEX(array,row\_num,[column\_num])

'Returns a value or reference of the cell at the intersection of a particular row and column, in a given range

'5) MIN(number1,[number2])

'Returns the smallest number in a set of values. Ignores logical values and text

'6) Row (reference)

'Returns the rownumber of a reference

'-> How to create an array formula:

'1) Copy (Ctrl + c) and paste (Ctrl + v) array formula to formula bar.

'2) Press and hold Ctrl + Shift.

'3) Press Enter once.

'4) Release all keys.

'-> How to remove #num errors:

'1) Array formula: =IFERROR(array\_formula, "")

Sub Return\_multiple\_values\_vertically()

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim SRC\_RNG As Range

Dim VALUES\_STR As String

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

VALUES\_STR = "Pen,1.5,Eraser,2,Paper,1.7,Pen,1.7,Paper clip,3,"

i = 1

For k = 1 To 5

j = InStr(i, VALUES\_STR, ",")

.Cells(1 + k, 2) = Mid(VALUES\_STR, i, j - i)

i = j + 1: j = InStr(i, VALUES\_STR, ",")

.Cells(1 + k, 3) = Mid(VALUES\_STR, i, j - i)

i = j + 1

Next k

Set SRC\_RNG = Range(.Cells(2, 2), .Cells(6, 3))

.Cells(8, 2) = "Pen"

For k = 1 To 3

'This array formula is entered in cell C8. Then copy cell C8 and paste to cells below.

'Array formula in C8:

'=INDEX($C$2:$C$6, SMALL(IF($B$8=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, ""), ROW(A1)))

.Cells(8 + k - 1, 3).FormulaArray = "=INDEX(" & SRC\_RNG.Address & \_

",SMALL(IF(($B$8=$B$2:$B$6),ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1," & \_

"""" & """" & "),ROW(A" & CStr(k) & ")),2)"

'or use this regular formula:

'=INDEX($C$2:$C$5, SMALL(INDEX(($B$8=$B$2:$B$6)\*(MATCH(ROW($B$2:$B$6), ROW($B$2:$B$6)))+

'($B$8<>$B$2:$B$6)\*1048577, 0, 0), ROW(A1)))

.Cells(8 + k - 1, 4).Formula = "=INDEX($C$2:$C$5, SMALL(INDEX(($B$8=$B$2:$B$6)\*(MATCH(ROW($B$2:$B$6)," & \_

"ROW($B$2:$B$6)))+($B$8<>$B$2:$B$6)\*1048577, 0, 0), ROW(A" & CStr(k) & ")))"

Next k

'Explaining the array formula for returning values vertically!

'Step 1 - Identify cells equal to the criterion:

'=INDEX($C$2:$C$6, SMALL(IF($B$8=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, ""), ROW(A1)))

'= (equal sign) is a comparison operator and checks if criterion ($B$8) is equal to values in array

'($B$2:$B$6). This operator is not case sensitive.

'$B$8=$B$2:$B$6

'becomes: "Pen"={"Pen", "Eraser", "Paper", "Pen", "Paper Clip"}

'becomes: {"Pen"="Pen", "Pen"="Eraser", "Pen"="Paper"; "Pen"="Pen", "Pen"="Paper Clip"}

'becomes: {TRUE, FALSE, FALSE, TRUE, FALSE}

'Step 2 - Create array containing row numbers

'=INDEX($C$2:$C$6, SMALL(IF($B$8=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, ""), ROW(A1)))

'ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1

'becomes: {2,3,4,5,6} - MIN({2,3,4,5,6})+1

'becomes: {2,3,4,5,6} - 2+1

'becomes: {2,3,4,5,6} - 1

'becomes: {1,2,3,4,5}

'Step 3 - Filter row numbers equal to criterion

'=INDEX($C$2:$C$6, SMALL(IF({TRUE,FALSE,FALSE,TRUE,FALSE},{1,2,3,4,5} , ""), ROW(A1)))

'becomes: INDEX($C$2:$C$6, SMALL({1,"","",4,""}, ROW(A1)))

'Step 4 - Return the k-th smallest row number

'=INDEX($C$2:$C$6, SMALL(IF($B$8=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, ""), ROW(A1)))

'SMALL(IF($B$8=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, ""), ROW(A1))

'becomes: SMALL({1,"","",4,""}, ROW(A1))

'This part of the formula returns the k-th smallest number in the array (1,"","",4,"")

'To calcualte the k-th smallest value I am using ROW(A1) to create the number 1.

'When the formula in cell C8 is copied to cell C9, ROW(A1) changes to ROW(A2). ROW(A2) is 2.

'In Cell C8: =INDEX($C$2:$C$6, SMALL((1,"","",4,""), ROW(A1)))

'=INDEX($C$2:$C$6, SMALL((1,"","",4,""), 1))

'The smallest number in array (1,"","",4,"") is 1.

'In Cell C9: =INDEX($C$2:$C$6, SMALL((1,"","",4,""), ROW(A2)))

'=INDEX($C$2:$C$6, SMALL((1,"","",4,""), 2))

'The second smallest number in array (1,"","",4,"") is 4.

'Step 5 - Return value in range

'INDEX(array,row\_num,[column\_num])

'Returns a value or reference of the cell at the intersection of a particular row and column, in a given range

'In Cell C8: =INDEX($C$2:$C$6,1)

'becomes =INDEX({1.5,2,1,1.7,3}, 1)

'and returns $1.50

'In Cell C9: =INDEX($C$2:$C$6,4) is $1,70

.Cells(13, 2) = "Pen"

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

For k = 1 To 3

.Cells(13 + k - 1, 3).FormulaArray = "=INDEX(" & SRC\_RNG.Address & ",SMALL(IF(($B$13=$B$2:$B$6)," & \_

"ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1," & """" & """" & "),ROW(A" & CStr(k) & ")),2)"

'Dynamic Range

.Cells(13 + k - 1, 4).FormulaArray = "=INDEX(" & SRC\_RNG.Address & ",SMALL(IF(($B$13=INDEX(" & SRC\_RNG.Address & \_

",0,1)),ROW(INDEX(" & SRC\_RNG.Address & ",0,1))-MIN(ROW(INDEX(" & SRC\_RNG.Address & \_

",0,1)))+1," & """" & """" & "),ROW(A" & CStr(k) & ")),2)"

Next k

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

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

Sub Return\_multiple\_values\_horizontally()

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim CHR\_STR As String

Dim SRC\_RNG As Range

Dim VALUES\_STR As String

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

VALUES\_STR = "Pen,1.5,Eraser,2,Paper,1.7,Pen,1.7,Paper clip,3,"

i = 1

For k = 1 To 5

j = InStr(i, VALUES\_STR, ",")

.Cells(1 + k, 2) = Mid(VALUES\_STR, i, j - i)

i = j + 1: j = InStr(i, VALUES\_STR, ",")

.Cells(1 + k, 3) = Mid(VALUES\_STR, i, j - i)

i = j + 1

Next k

Set SRC\_RNG = Range(.Cells(2, 2), .Cells(6, 3))

.Cells(8, 2) = "Pen"

For k = 1 To 3

CHR\_STR = IIf(k = 1, "A1", IIf(k = 2, "B1", "C1"))

'This array formula is entered in cell C9. Then copy cell C9 and paste to the right.

'Array formula in C9:

'=INDEX($C$2:$C$6, SMALL(IF($B$9=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, ""), COLUMN(A1)))

.Cells(8, 2 + k).FormulaArray = "=INDEX($C$2:$C$6, SMALL(IF($B$8=$B$2:$B$6, ROW($B$2:$B$6)-MIN(ROW($B$2:$B$6))+1, " \_

& """" & """" & "), COLUMN(" & CHR\_STR & ")))"

'or use this regular formula:

'=INDEX($C$2:$C$5, SMALL(INDEX(($B$9=$B$2:$B$6)\*(MATCH(ROW($B$2:$B$6), ROW($B$2:$B$6)))+($B$9<>$B$2:$B$6)\*1048577, 0, 0), COLUMN(A1)))

.Cells(9, 2 + k).Formula = "=INDEX($C$2:$C$5, SMALL(INDEX(($B$8=$B$2:$B$6)\*(MATCH(ROW($B$2:$B$6), " & \_

"ROW($B$2:$B$6)))+($B$8<>$B$2:$B$6)\*1048577, 0, 0), COLUMN(" & CHR\_STR & ")))"

Next k

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

Sub Return\_multiple\_records()

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim L As Integer

Dim NROWS As Integer

Dim NCOLUMNS As Integer

Dim CHR\_STR As String

Dim SRC\_RNG As Range

Dim VALUES\_STR As String

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

VALUES\_STR = "id,name,status|123,ANNA,active|124,jhun,separated|125,liza,active|" & \_

"129,roy,separated|789,mary,separated|123,anna,separated|"

For i = 1 To Len(VALUES\_STR)

If Mid(VALUES\_STR, i, 1) = "|" Then: NROWS = NROWS + 1

Next i

NCOLUMNS = 3

i = 1

For k = 1 To NROWS

For L = 1 To NCOLUMNS

If L < NCOLUMNS Then

j = InStr(i, VALUES\_STR, ",")

Else

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

End If

.Cells(k, L) = Mid(VALUES\_STR, i, j - i)

i = j + 1

Next L

Next k

Set SRC\_RNG = Range(.Cells(1, 1), .Cells(NROWS, NCOLUMNS))

.Cells(9, 1) = "enter id:"

.Cells(9, 2) = 123

For i = 1 To 3

For j = 1 To 3

CHR\_STR = IIf(j = 1, "A" & CStr(i), IIf(j = 2, "B" & CStr(i), "C" & CStr(i)))

'Array formula in cell A10:

'=INDEX($A$2:$C$7, SMALL(IF($B$9=$A$2:$A$7, ROW($A$2:$A$7)-MIN(ROW($A$2:$A$7))+1, ""), ROW(A1)),COLUMN(A1))

'Copy cell A10 and paste to cell range B10:C10. Then copy A10:C10 and paste to cell range A11:C12.

.Cells(9 + i, j).FormulaArray = "=INDEX($A$2:$C$7, SMALL(IF($B$9=$A$2:$A$7, " & \_

"ROW($A$2:$A$7)-MIN(ROW($A$2:$A$7))+1, " & """" & \_

"""" & "), ROW(" & CHR\_STR & ")),COLUMN(" & CHR\_STR & "))"

'or use this regular formula:

'=INDEX($A$2:$C$7, SMALL(INDEX(($B$9=$A$2:$A$7)\*(MATCH(ROW($A$2:$A$7), ROW($A$2:$A$7)))+($B$9<>$A$2:$A$7)\*

'1048577, 0, 0),ROW(A1)),COLUMN(A1))

.Cells(14 + i, j).Formula = "=INDEX($A$2:$C$7, SMALL(INDEX(($B$9=$A$2:$A$7)\*(MATCH(ROW($A$2:$A$7), " & \_

"ROW($A$2:$A$7)))+($B$9<>$A$2:$A$7)\*1048577, 0, 0),ROW(" & CHR\_STR & ")),COLUMN(" & \_

CHR\_STR & "))"

Next j

Next i

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

'The example below shows how to extract selected data from a range without Excel VLOOKUP, so that the three best

'scores for each individual could be extracted from a two column Excel table listing multiple individuals,

'each with several different scores.

'-> Excel Functions in this Example:

'1) IF(logical\_test;[value\_if:true];[value\_if\_false])

'Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE

'2) SMALL(array,k)

'Returns the k-th smallest number in this data set.

'3) LARGE(array,k)

'Returns the k-th largest number in this data set.

'4) INDEX(array,row\_num,[column\_num])

'Returns a value or reference of the cell at the intersection of a particular row and column, in a given range

'5) MIN(number1,[number2])

'Returns the smallest number in a set of values. Ignores logical values and text

'6) Row (reference)

'Returns the rownumber of a reference

'-> How to create an array formula:

'1) Copy (Ctrl + c) and paste (Ctrl + v) array formula to formula bar.

'2) Press and hold Ctrl + Shift.

'3) Press Enter once.

'4) Release all keys.

'-> How to remove #num errors:

'1) Array formula: =IFERROR(array\_formula, "")

Sub Extracting\_selected\_data\_with\_array\_functions()

Dim i As Long

Dim j As Long

Dim SROW As Long

Dim SCOLUMN As Long

Dim CHR\_STR As String

Dim DST\_RNG As Range

Dim TEMP\_VAL As Double

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

SROW = 5: SCOLUMN = 2 'Starting at B5

.Cells(SROW, 2) = "Name"

.Cells(SROW, 3) = "Score"

For i = 1 To 100

TEMP\_VAL = Rnd

.Cells(SROW + i, SCOLUMN) = IIf(TEMP\_VAL < 0.33, "Thomas", IIf(TEMP\_VAL < 0.66, "Richard", "Harold"))

.Cells(SROW + i, SCOLUMN + 1) = WorksheetFunction.RandBetween(70, 80)

Next i

Set DST\_RNG = Range(.Cells(SROW + 1, SCOLUMN), .Cells(SROW + 100, SCOLUMN + 1))

'If the names are in B6:B105 the scores are in C6:C105 and a selected name is in E6 then:

'(IF($B$6:$B$105=$H6,$C$6:$C$105,FALSE) will return an array of 100 items containing either a

'score for the selected name or FALSE.

i = SROW + 1

For j = SCOLUMN + 2 To SCOLUMN + 4

CHR\_STR = .Cells(i, SCOLUMN + 6).Address(False, True) ' "$H" & CStr(i)

i = i + 1

.Cells(SROW, j).Formula = "=" & CHR\_STR

Range(.Cells(SROW + 1, j), .Cells(SROW + 100, j)).FormulaArray = \_

"=IF(" & DST\_RNG.Columns(1).Address & "=" & CHR\_STR & "," & DST\_RNG.Columns(2).Address & ",FALSE)"

Next j

.Cells(SROW + 0, SCOLUMN + 6) = "Best 3 Scores"

.Cells(SROW + 1, SCOLUMN + 6) = "Thomas"

.Cells(SROW + 2, SCOLUMN + 6) = "Richard"

.Cells(SROW + 3, SCOLUMN + 6) = "Harold"

For j = 1 To 3

.Cells(SROW, SCOLUMN + 6 + j) = j

Next j

For i = 1 To 3

For j = 1 To 3

'You can then wrap the Large() function around that: =LARGE(IF($B$6:$B$105=$H6,$C$6:$C$105,FALSE),I$5)

'where F5 contains the rank you want. An example is shown below, listing the three best scores of

'Thomas, Richard and Harold:

.Cells(SROW + i, SCOLUMN + 6 + j).FormulaArray = \_

"=LARGE(IF(" & DST\_RNG.Columns(1).Address & "=" & \_

.Cells(SROW + i, SCOLUMN + 6 + j).Offset(0, -j).Address(False, True) & \_

"," & DST\_RNG.Columns(2).Address & ",FALSE)," & \_

.Cells(SROW + i, SCOLUMN + 6 + j).Offset(-i, 0).Address(True, False) & ")"

'Note that because this is an array formula it must be entered by pressing ñCtrl-Shift-Enterî. If

'you just press ñEnterî it returns 0.

'The working of the formula is shown in D6:F105, with the results of the formula:

'(IF($B$6:$B$105=$H6,$C$6:$C$105,FALSE)

Next j

Next i

'Note that the array includes one entry for every row of the original table, containing either a

'score for Thomas, or FALSE if the row is data for another person. This is not a problem for use

'with the Large() function, but if a continuous list of scores for reach individual is required

'then further work is required.

'In many cases simply sorting the table, using the Excel filter functions, or using a pivot table

'would be the easiest way to go, but if an automatically updating array formula is required it is

'possible to wrap some additional functions around the solution presented above:

i = SROW + 1

For j = SCOLUMN + 11 To SCOLUMN + 21 Step 4

CHR\_STR = .Cells(i, SCOLUMN + 6).Address(False, True) ' "$H" & CStr(i)

i = i + 1

'IF($H6=$B$6:$B$105,ROW($C$6:$C$105)-ROW($C$6)+1,FALSE)

'will return an array of row numbers for rows containing scores for Thomas, with FALSE in the

'other rows.

.Cells(SROW, j + 0).Formula = "=" & CHR\_STR

Range(.Cells(SROW + 1, j + 0), .Cells(SROW + 100, j + 0)).FormulaArray = \_

"=IF(" & CHR\_STR & "=" & DST\_RNG.Columns(1).Address & ",ROW(" & \_

DST\_RNG.Columns(2).Address & ")-ROW(" & DST\_RNG.Columns(2).Cells(1, 1).Address & \_

")+1,FALSE)"

'=SMALL(IF($H6=$B$6:$B$105,ROW($C$6:$C$105)-ROW($C$6)+1,FALSE),ROW(B1:B99))

'Extracts the row numbers for Thomas from this array, without the FALSE values.

.Cells(SROW, j + 1).Formula = "=" & CHR\_STR

Range(.Cells(SROW + 1, j + 1), .Cells(SROW + 100, j + 1)).FormulaArray = \_

"=SMALL(IF(" & CHR\_STR & "=" & DST\_RNG.Columns(1).Address & ",ROW(" & \_

DST\_RNG.Columns(2).Address & ")-ROW(" & DST\_RNG.Columns(2).Cells(1, 1).Address & \_

")+1,FALSE),ROW(" & Range(.Cells(1, SCOLUMN), .Cells(100, SCOLUMN)).Address & "))"

'=INDEX($C$6:$C$105,SMALL(IF($H6=$B$6:$B$105,ROW($C$6:$C$105)-ROW($C$6)+1,FALSE),ROW(B1:B99)))

'extracts the actual score from those row numbers.

.Cells(SROW, j + 2).Formula = "=" & CHR\_STR

Range(.Cells(SROW + 1, j + 2), .Cells(SROW + 100, j + 2)).FormulaArray = \_

"=INDEX(" & DST\_RNG.Columns(2).Address & ",SMALL(IF(" & CHR\_STR & "=" & \_

DST\_RNG.Columns(1).Address & ",ROW(" & DST\_RNG.Columns(2).Address & \_

")-ROW(" & DST\_RNG.Columns(2).Cells(1, 1).Address & \_

")+1,FALSE),ROW(" & Range(.Cells(1, SCOLUMN), .Cells(100, SCOLUMN)).Address & ")))"

Next j

'In this case the first approach is not only much shorter, it is also much more convenient because

'before the Large() function can be used on the array returned by the second method the length of

'the array needs to be determined.

'Once again, remember all the formulas presented here must be entered as array formulas. Failure to

'enter as an array will sometimes result in an error message, but can return incorrect values, so

'beware.

'Remember that the basic technique is to replace the IF statement with an array multiplication:

'($B$6:$B$105=E5) returns an array of TRUE and FALSE values. When these are multiplied by

'the values in $C$6:$C$105 it returns an array of the values in column C or 0 (since TRUE and

'FALSE are treated as equal to 1 and 0 in numerical calculations), and then the Large

'function may be used directly on that array. This is a very powerful technique, with

'many applications, the only proviso being that it is not suitable where the data may

'include 0 values, since it will be impossible to differentiate between this value and

'the values that do not match the selection criteria.

'The other difference is that the Large function has also been applied as an array formula,

'so that rather than array entering the formula in a single cell and copying it to the two

'adjacent cells it must be array entered with the three adjacent cells selected. It will

'then return the top three scores for the chosen person.

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub

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

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

'Section 9. Writing UDF Array Formulas

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

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

'As we already notice Excel array formulas can do amazing things. They are like ordinary formulas except that you enter them with

'Control/Shift/Enter rather than just enter. So far we have seen two kinds of array formulae:

'i) Single cell array formulae are entered into a single cell, loop through their arguments (which are often calculated arguments)

'and return a single answer.

'ii) Multi-cell array formulae are entered into multiple cells and return an answer to each of the cells.

'With this power comes a cost: because array formulae are doing a lot of work they can be slow to calculate (particularly single-cell

'array formulas).

'We can break down the time taken by a VBA UDF into these components:

'i) Overhead time to call the UDF.

'ii) Time to fetch the data thats going to be used by the UDF.

'iii) Time to do the calculations.

'iv) Overhead time to return the answer(s).

'In lesson 3.9 we learned that there was quite a significant overhead on each VBA read and write call,

'so that its usually much faster to read and write large blocks of data at a time.

'So it sounds like a good idea to make your VBA UDF read as much data as possible in a single block and return data to Excel in as large

'a block as possible. Enter the Multi-cell array formula \_ it does exactly that \_ and also minimises the calling overhead \_ and often it

'can read the data once and re-use it lots of times.

'Lets create an array version of the VLOOKUP function: VLOOKUP\_FUNC(lookup\_value, table\_array, col\_index\_num, [h])

Function VLOOKUP\_FUNC(LOOK\_VAL As Range, \_

TABLE\_RNG As Range, \_

INDEX\_VAL As Integer, \_

Optional LAYOUT\_STR As String = "v")

'LAYOUT\_STR is optional:

' h= return values horizontally

' v= return values vertically

Dim i As Single

Dim NROWS As Single

Dim NCOLUMNS As Single

Dim TEMP\_ARR() As Variant

ReDim TEMP\_ARR(0 To 0)

For i = 1 To TABLE\_RNG.rows.count

If LOOK\_VAL = TABLE\_RNG.Cells(i, 1) Then

TEMP\_ARR(UBound(TEMP\_ARR)) = TABLE\_RNG.Cells(i, INDEX\_VAL)

ReDim Preserve TEMP\_ARR(UBound(TEMP\_ARR) + 1)

End If

Next i

If LAYOUT\_STR = "h" Then

NCOLUMNS = Range(Application.Caller.Address).Columns.count

For i = UBound(TEMP\_ARR) To NCOLUMNS

TEMP\_ARR(UBound(TEMP\_ARR)) = ""

ReDim Preserve TEMP\_ARR(UBound(TEMP\_ARR) + 1)

Next i

ReDim Preserve TEMP\_ARR(UBound(TEMP\_ARR) - 1)

VLOOKUP\_FUNC = TEMP\_ARR

Else

NROWS = Range(Application.Caller.Address).rows.count

For i = UBound(TEMP\_ARR) To NROWS

TEMP\_ARR(UBound(TEMP\_ARR)) = ""

ReDim Preserve TEMP\_ARR(UBound(TEMP\_ARR) + 1)

Next i

ReDim Preserve TEMP\_ARR(UBound(TEMP\_ARR) - 1)

VLOOKUP\_FUNC = Application.Transpose(TEMP\_ARR)

End If

End Function

'The scenario below is that we want to return multiple values vertically or horizontally.

'the function returns an array of values. In many real-life cases using multi-cell array UDFs

'can be the fastest way to calculate. Converting a conventional UDF to a multi-cell array UDf

'is straightforward.

Sub Return\_multiple\_values\_vertically\_or\_horizontally() 'Testing Function VLOOKUP\_FUNC (below)

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim SRC\_RNG As Range

Dim VALUES\_STR As String

Dim SRC\_WSHEET As Worksheet

Call EXCEL\_TURN\_OFF\_EVENTS\_FUNC

Set SRC\_WSHEET = WSHEET\_ADD\_FUNC(PARSE\_CURRENT\_TIME\_FUNC("\_"), ActiveWorkbook)

With SRC\_WSHEET

'.Cells.Clear 'Remove all values/formulas in sheet

VALUES\_STR = "Pen,1.5,Eraser,2,Paper,1.7,Pen,1.7,Paper clip,3,"

i = 1

For k = 1 To 5

j = InStr(i, VALUES\_STR, ",")

.Cells(1 + k, 2) = Mid(VALUES\_STR, i, j - i)

i = j + 1: j = InStr(i, VALUES\_STR, ",")

.Cells(1 + k, 3) = Mid(VALUES\_STR, i, j - i)

i = j + 1

Next k

Set SRC\_RNG = Range(.Cells(2, 2), .Cells(6, 3))

.Cells(8, 2) = "Return values vertically"

.Cells(9, 2) = "Pen"

''Array formula in cell C9:C11: =VLOOKUP\_FUNC(B9, B2:C6, 2)

'How to enter custom function array formula in Excel

'Select cell range C9:C11

'Type above custom function

'Press and hold Ctrl + Shift

'Press Enter once

'Release all keys

Range(.Cells(9, 3), .Cells(11, 3)).FormulaArray = "=VLOOKUP\_FUNC(" & \_

.Cells(9, 2).Address & "," & SRC\_RNG.Address & ",2)"

.Cells(13, 2) = "Return values horizontally"

.Cells(14, 2) = "Pen"

.Cells(15, 2) = "Paper"

.Cells(16, 2) = "Eraser"

For k = 1 To 3

'Array formula in cell C14:D14:

'=vbaVlookup(B14, $B$2:$C$6, 2, "h")

'How to enter custom function array formula

'Select cell range C14:D14

'Type above custom function

'Press and hold Ctrl + Shift

'Press Enter once

'Release all keys

Range(.Cells(13 + k, 3), .Cells(13 + k, 5)).FormulaArray = \_

"=VLOOKUP\_FUNC(" & .Cells(13 + k, 2).Address(False, True) & "," & \_

SRC\_RNG.Address & ",2," & """" & "h" & """" & ")"

'How to copy array formula to the next row

'Select cell range C14:D14

'Copy cell range

'Select cell range C15:D15

'Paste -> .Cells(13 + k, 2).Address(False, True)

Next k

End With

Call EXCEL\_TURN\_ON\_EVENTS\_FUNC

End Sub