Sub AddKeys()

' Paul Beverley - Version 08.07.20

' Adds keybindings to macros

' Sub CustomKeys = Shift-Alt-K

' Sub AccentAlyse = Shift-Alt-A

' Sub DocAlyse = Shift-Alt-D

' Sub FindBack = Alt-Left

' Sub FindBackCase = Shift-Alt-Left

' Sub FindFwd = Alt-Right

' Sub FindFwdCase = Shift-Alt-Right

' Sub FRedit = Shift-Alt-F

' Sub HyphenAlyse = Shift-Alt-H

' Sub InstantFindDown = Shift-Alt-Down

' Sub InstantFindUp = Shift-Alt-Up

' Sub IStoIZ = Shift-Alt-Z

' Sub IZIScount = Shift-Alt-I

' Sub IZtoIS = Shift-Alt-S

' Sub MultiSwitch = Alt-Q

' Sub CopyTextSimple = Alt-C

' Sub MultiFileText = Alt-T

' Sub ProperNounAlyse = Shift-Alt-P

' Sub SpellingErrorHighlighter = Shift-Alt-E

' Sub SpellAlyse = Shift-Alt-L

' Sub UKUSCount = Shift-Alt-U

' Sub WhatChar = Shift-Alt-Slash

' Sub WordPairAlyse = Shift-Alt-W

' Sub MacroLauncher = Shift-Alt-M

keyCount = 0

For Each par In ActiveDocument.Paragraphs

myLine = Replace(par.Range.Text, vbCr, "")

myLine = Replace(myLine, " ", "")

If Left(myLine, 6) = "EndSub" Then

If keyCount > 0 Then

MsgBox keyCount & " shortcut keys programmed."

Else

Beep

MsgBox "Please ensure that 'The\_Starter\_Macros' document is" & vbCr \_

& "still open on screen, and that the cursor" & vbCr \_

& "is in that file. Now re-run this macro."

End If

Exit Sub

End If

gotOne = False

If Left(myLine, 4) = "'Sub" Then myLine = Mid(myLine, 5): gotOne = True

eqPos = InStr(myLine, "=")

If gotOne And eqPos > 0 Then

keyCount = keyCount + 1

myMacroName = Left(myLine, eqPos - 1)

myKeyCode = 0

myKeyText = Mid(myLine, eqPos + 1)

i = Len(myKeyText)

myKeyText = Replace(myKeyText, "Ctrl-", "")

If Len(myKeyText) < i Then myKeyCode = myKeyCode + wdKeyControl

i = Len(myKeyText)

myKeyText = Replace(myKeyText, "Shift-", "")

If Len(myKeyText) < i Then myKeyCode = myKeyCode + wdKeyShift

i = Len(myKeyText)

myKeyText = Replace(myKeyText, "Alt-", "")

If Len(myKeyText) < i Then myKeyCode = myKeyCode + wdKeyAlt

If Len(myKeyText) = 1 Then

myKeyCode = myKeyCode + Asc(myKeyText)

Else

If Asc(myKeyText) = Asc("F") Then

aCode = 111 + Val(Replace(myKeyText, "F", ""))

Else

Select Case myKeyText

Case "Up": aCode = vbKeyUp ' The Up Cursor key

Case "Down": aCode = vbKeyDown ' The Down Cursor key

Case "Right": aCode = vbKeyRight ' The Right Cursor key

Case "Left": aCode = vbKeyLeft ' The Left Cursor key

Case "BackSingleQuote": aCode = wdKeyBackSingleQuote ' The ` key

Case "BackSlash": aCode = wdKeyBackSlash ' The \ key

Case "Backspace": aCode = wdKeyBackspace ' The Backspace key

Case "CloseSquareBrace": aCode = wdKeyCloseSquareBrace ' The ] key

Case "Comma": aCode = wdKeyComma ' The , key

Case "Delete": aCode = wdKeyDelete ' The Delete key

Case "End": aCode = wdKeyEnd ' The EndD key

Case "Equals": aCode = wdKeyEquals ' The = key

Case "Home": aCode = wdKeyHome ' The HOME key

Case "Hyphen": aCode = wdKeyHyphen ' The - key

Case "Insert": aCode = wdKeyInsert ' The Insert key

Case "NumAdd": aCode = wdKeyNumericAdd ' The + key on the keypad

Case "NumDecimal": aCode = wdKeyNumericDecimal

' The . key on the keypad

Case "NumDivide": aCode = wdKeyNumericDivide

' The / key on the keypad

Case "NumMultiply": aCode = wdKeyNumericMultiply

' The \* key on the keypad

Case "NumSubtract": aCode = wdKeyNumericSubtract

' The - key on the keypad

Case "OpenSquareBrace": aCode = wdKeyOpenSquareBrace

' The [ key

Case "PageDown": aCode = wdKeyPageDown ' The Page Down key

Case "PageUp": aCode = wdKeyPageUp ' The Page Up key

Case "Period": aCode = wdKeyPeriod ' The key

Case "Return": aCode = wdKeyReturn ' The Enter or Return key

Case "SemiColon": aCode = wdKeySemiColon ' The ; key

Case "SingleQuote": aCode = wdKeySingleQuote ' The ' key

Case "Slash": aCode = wdKeySlash ' The / key

Case "Spacebar": aCode = wdKeySpacebar ' The Spacebar key

Case Else: MsgBox ("Error - Can't find: " & myLine): Exit Sub

End Select

End If

myKeyCode = myKeyCode + aCode

End If

KeyBindings.Add KeyCode:=myKeyCode, KeyCategory:=wdKeyCategoryMacro, \_

Command:=myMacroName

End If

Next par

If keyCount > 0 Then

MsgBox keyCount & " shortcut keys programmed."

Else

Beep

MsgBox "Please reopen the Word file containing" & vbCr \_

& "all the macros: TheStarterMacros."

End If

End Sub

Sub AccentAlyse()

' Paul Beverley - Version 06.02.20

' Analyses all the words that contain an accent

' These are the accents to watch out for

allAccents = "áÁàÀâÂäÄÃãÅåçÇéÉèÈêÊëËíÍìÌîÎñÑóÓòÒôÔöÖõÕøØßúÚùÙûÛüÜýÝÿŸ"

' For including 'Central European' characters

addExtraCharacters = True

fromUnicode = 256

toUnicode = 382

' Minimum word length

minLength = 3

Set FUT = ActiveDocument

doingSeveralMacros = (InStr(FUT.Name, "zzTestFile") > 0)

If doingSeveralMacros = False Then

myResponse = MsgBox(" AccentAlyse" & vbCr & vbCr & \_

"Analyse this document?", vbQuestion \_

+ vbYesNoCancel, "AccentAlyse")

If myResponse <> vbYes Then Exit Sub

End If

If addExtraCharacters = True Then

For i = fromUnicode To toUnicode

allAccents = allAccents & ChrW(i)

Next i

End If

myLead = " . . . "

Set rng = ActiveDocument.Content

Documents.Add

Set resultDoc = ActiveDocument

Set res = ActiveDocument.Content

Documents.Add

Set testDoc = ActiveDocument

Selection.Text = rng.Text

Selection.HomeKey Unit:=wdStory

Selection.TypeText "Finding accented words..." & vbCr \_

& vbCr & vbCr

Selection.Start = 0

Selection.range.Style = ActiveDocument.Styles(wdStyleHeading1)

Selection.Font.Underline = True

Set rng = testDoc.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[" & allAccents & "]"

.Font.Underline = False

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

Do While rng.Find.Found = True

DoEvents

rng.Expand wdWord

myEnd = rng.End

If InStr(" " & ChrW(8217), Right(rng.Text, 1)) \_

> 0 Then rng.End = rng.End - 1

If InStr(" " & ChrW(8217), Right(rng.Text, 1)) \_

> 0 Then rng.End = rng.End - 1

myAccWord = rng.Text

If Len(myAccWord) >= minLength Then

Set rng2 = testDoc.Content

myTot = rng2.End

DoEvents

With rng2.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = myAccWord

.Wrap = wdFindContinue

.MatchWholeWord = True

.Replacement.Text = "^&!"

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

myCount = testDoc.Content.End - myTot

WordBasic.EditUndo

DoEvents

Selection.End = 0

With rng2.Find

.Replacement.Text = "^&"

.Replacement.Font.Underline = True

.MatchWholeWord = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

DoEvents

' Find an accent-different word

findWord = ""

myAltWord = ""

For i = 1 To Len(myAccWord)

myChar = Mid(myAccWord, i, 1)

If InStr(allAccents, myChar) > 0 Then

findWord = findWord & "^$"

Else

findWord = findWord & myChar

End If

Next i

testDoc.Content.Font.StrikeThrough = False

With rng2.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = findWord

.Font.Underline = False

.Font.StrikeThrough = False

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = False

.Execute

End With

Do While rng2.Find.Found = True

DoEvents

myAltWord = rng2.Text

Set rng3 = testDoc.Content

myTot = rng3.End

With rng3.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = myAltWord

.Font.Underline = False

.Wrap = wdFindContinue

.MatchWholeWord = True

.Replacement.Text = "^&!"

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

DoEvents

myAltCount = testDoc.Content.End - myTot

WordBasic.EditUndo

DoEvents

Selection.End = 0

DoEvents

With rng3.Find

.ClearFormatting

.Replacement.ClearFormatting

.Replacement.Text = "^&"

.Replacement.Font.StrikeThrough = True

.MatchWholeWord = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

DoEvents

If InStr(res.Text, myAltWord & myLead) = 0 Then

myLine = myAltWord & myLead & Trim(Str(myAltCount)) & vbCr

res.InsertAfter myLine

ActiveDocument.Paragraphs(2).range.Text = myLine

ActiveDocument.Paragraphs(2).range.Font.StrikeThrough = True

End If

rng2.Start = myEnd

With rng2.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = findWord

.Font.Underline = False

.Font.StrikeThrough = False

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = False

.Execute

End With

Loop

If myAltWord > "" Then

myLine = myAccWord & myLead & Trim(Str(myCount)) & vbCr

res.InsertAfter myLine

ActiveDocument.Paragraphs(1).range.Text = myLine

End If

End If

rng.Start = myEnd

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[" & allAccents & "]"

.Font.Underline = False

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

Loop

testDoc.Close SaveChanges:=False

resultDoc.Activate

Set rng = ActiveDocument.Content

rng.Sort SortOrder:=wdSortOrderAscending

Selection.HomeKey Unit:=wdStory

Selection.TypeText "Accent Use" & vbCr

ActiveDocument.Paragraphs(1).Style = ActiveDocument.Styles(wdStyleHeading1)

If doingSeveralMacros = False Then

Beep

Else

FUT.Activate

End If

End Sub

Sub WhatChar()

' Paul Beverley - Version 10.08.20

' Show ASCII and unicode and character names

' To use voice, "uncomment" this line:

' Set speech = New SpVoice

' Decide where to show results

showOnStatusbar = False

showInMessageBox = True

useVoice = False

' If the macro gives an error on locked files, make this True

lockedFile = False

' lockedFile = True

' If the character is a space, don't show a message box.

justBeepForSpace = False

' If you also want to know what the hex of the ANSI code is:

showHexANSI = True

' If you also want to use this code in FRedit:

prepareFReditCode = True

If showOnStatusbar + showInMessageBox + useVoice = 0 Then \_

Beep: MsgBox "Alter options in macro code to show results.": \_

Exit Sub

charSelected = (Selection.Start <> Selection.End)

If AscW(Selection) = 32 And justBeepForSpace Then Beep: Exit Sub

CR2 = vbCr & vbCr

If Selection.Start = Selection.End Then Selection.MoveEnd wdCharacter, 1

If lockedFile = False Then Selection.Range.Revisions.AcceptAll

ansicode = Asc(Selection)

uCode = Val(Dialogs(wdDialogInsertSymbol).CharNum)

If Asc(ansicode) = 63 And uCode <> 63 Then

ansiBit = "ANSI: ???"

Else

ansiBit = ">>>>>> ANSI: " & Str(ansicode)

If showHexANSI = True Then ansiBit = ansiBit & \_

" (hex " & Hex(ansicode) & ")"

End If

hexCode = Replace(Hex(uCode), "FFFF", "")

ucodeBit = vbCr & "Unicode: " & Str(uCode) & " (hex " & \_

hexCode & ")"

If Selection.Font.Name <> \_

ActiveDocument.Styles(wdStyleNormal).Font.Name Then

fontBit = vbCr & "Font: " & Selection.Font.Name

Else

fontBit = ""

End If

fntSize = Selection.Font.Size

normalSize = ActiveDocument.Styles(wdStyleNormal).Font.Size

If normalSize <> fntSize Then \_

fontBit = fontBit & " Font Size: " & Str(fntSize)

HexBit = ""

If uCode < 0 Then

HexBit = vbCr & vbCr & "For FRedit, use Find: <&H" & hexCode & ">"

If prepareFReditCode = True Then

Selection.Collapse wdCollapseStart

startCode = Selection.Start

Selection.TypeText "<&H" & hexCode & ">|"

Selection.Start = startCode

Selection.Cut

End If

End If

' To correct for Mac codes

Select Case ansicode

Case 2: extraBit = "Foot note end note marker"

Case 32: extraBit = "Ordinary space"

Case 34: extraBit = "Straight double quote"

Case 39: extraBit = "Straight apostrophe"

Case 40: If Selection.Font.Name = "Symbol" Then extraBit = "Funny Symbol font character!"

Case 45: extraBit = "Ordinary hyphen"

Case 46: extraBit = "Full point"

Case 48: extraBit = "Number 0"

Case 49: extraBit = "Number 1"

Case 50: extraBit = "Number 2"

Case 51: extraBit = "Number 3"

Case 58: extraBit = "Colon"

Case 59: extraBit = "Semicolon"

Case 63: extraBit = "Question mark"

Case 73: extraBit = "Uppercase I(eye)"

Case 76: extraBit = "Uppercase L"

Case 79: extraBit = "Uppercase O"

Case 88: extraBit = "Uppercase X"

Case 96: extraBit = "Back tick"

Case 105: extraBit = "Lowercase i (eye)"

Case 108: extraBit = "Lowercase l (el)"

Case 111: extraBit = "Lowercase o"

Case 120: extraBit = "Lowercase x"

Case 150: extraBit = "En dash"

Case 151: extraBit = "Em dash"

Case 174: extraBit = "Registered trademark"

Case 176: extraBit = "Proper degree symbol"

Case Else: extraBit = ""

End Select

' The unicode information will overwrite the ANSI

Select Case uCode

Case 9: extraBit = "Tab character"

Case 11: extraBit = "Manual line break"

Case 13: extraBit = "Just an ordinary paragraph end"

Case 30: extraBit = "Non-breaking hyphen"

Case 31: extraBit = "Optional hyphen"

Case 124: extraBit = "Vertical bar"

Case 160: extraBit = "Non-breaking space"

Case 176: extraBit = "Proper degree symbol"

Case 178: extraBit = "Dodgy squared symbol!"

Case 179: extraBit = "Dodgy cubed symbol!"

Case 180: extraBit = "Dodgy apostrophe!"

Case 186: extraBit = "Masculine ordinal"

Case 215: extraBit = "Proper multiply symbol"

Case 937: extraBit = "Omega"

Case 956: extraBit = "Mu = micro"

Case 8194: extraBit = "En space"

Case 8195: extraBit = "Em space"

Case 8201: extraBit = "Thin space"

Case 8216: extraBit = "Single open curly quote"

Case 8217: extraBit = "Single close curly quote = apostrophe"

Case 8211: extraBit = "En dash"

Case 8212: extraBit = "Em dash"

Case 8220: extraBit = "Double open curly quote"

Case 8221: extraBit = "Double close curly quote"

Case 8222: extraBit = "German open curly quote"

Case 8226: extraBit = "Ordinary bullet"

Case 8230: extraBit = "Ellipsis"

Case 8242: extraBit = "Unicode: single prime"

Case 8243: extraBit = "Unicode: double prime"

Case 8249: extraBit = "French open quote"

Case 8250: extraBit = "French close quote"

Case 8722: extraBit = "Proper minus sign"

End Select

If extraBit > "" Then extraBit = " >>>> " & extraBit

If Selection.Font.Superscript = True Then

extraBit = extraBit & " >>>> superscripted"

End If

If Selection.Font.Subscript = True Then

extraBit = extraBit & " >>>> subscripted"

End If

If Selection.Font.Italic = True Then

extraBit = extraBit & " >>>> italic"

End If

' This isn't a 'funny' Symbol font, undo the Revisions.AcceptAll

If lockedFile = False And uCode > 0 Then WordBasic.EditUndo

' Show on the status bar

s = " "

If showOnStatusbar = True Then \_

StatusBar = s & s & s & s & ansiBit & " " & extraBit \_

& " " & ucodeBit & " " & fontBit & HexBit

If useVoice And extraBit > "" And charSelected = False Then \_

speech.Speak extraBit, SVSFPurgeBeforeSpeak

' Display in a message box

If (useVoice And charSelected) Or showInMessageBox \_

Or extraBit = "" Or uCode < 0 Then \_

MsgBox ansiBit & extraBit & vbCr & ucodeBit & vbCr & fontBit \_

& HexBit, , "WhatChar"

Selection.Collapse wdCollapseStart

End Sub

Sub CustomKeys()

' Paul Beverley - Version 03.02.10

' Open the Customize Keyboard dialogue box

With Dialogs(wdDialogToolsCustomizeKeyboard)

.Category = 2

.Show

End With

End Sub

Sub ProperNounAlyse()

' Paul Beverley - Version 29.04.25

' Analyses similar proper nouns

minLengthCheck = 4

includeAcronyms = True

' myLanguage = "English(United Kingdom)"

myLanguage = "English(United States)"

ignoreWords = "Also Being The This There Those Their They Then These That" & \_

" Ever Even Four From Have When Where Will While Were Whole Well Would"

similarChars = "bb,b; b,p; sch,sh; ch,sh; c,k; ph,f; ss,z; s,z;" & \_

" mp,m; ll,l; nn,n; nd,n; nt,n;"

' With non-English languages, you might need to make this False

ignorePlurals = True

keepWholeList = False

keepWholeList = True

myScreenOff = True

CR = vbCr: CR2 = CR & CR

Set FUT = ActiveDocument

doingSeveralMacros = (InStr(FUT.Name, "zzTestFile") + \_

InStr(FUT.Name, "Document") > 0)

If doingSeveralMacros = False Then

myResponse = MsgBox(" ProperNounAlyse" & CR2 & \_

"Analyse this document?", vbQuestion \_

+ vbYesNoCancel, "ProperNounAlyse")

If myResponse <> vbYes Then Exit Sub

End If

If FUT.Content.Words.Count > 50000 Then

useQuickSort = True

Else

useQuickSort = False

End If

If myScreenOff = True Then

Application.ScreenUpdating = False

On Error GoTo ReportIt

End If

myDummy = ChrW(222)

For i = 1 To 100

spcs = " " & spcs

Next i

checkFinalLetters = False

checkFinalLetters = True

' Grey on word only

thisHighlight = wdGray25

doMissingLetter = True

' doMissingLetter = False

' Bold And blue

switchTest = True

' switchTest = False

' double strikethrough

doSimilarLetters = True

' doSimilarLetters = False

' various highlight colours + underline

doVowelTest = True

' doVowelTest = False

' various highlight colours + italic

' These last two tests cycle through these colours:

maxCol = 6

ReDim myCol(maxCol) As Integer

myCol(1) = wdYellow

myCol(2) = wdBrightGreen

myCol(3) = wdTurquoise

myCol(4) = wdRed

myCol(5) = wdPink

myCol(6) = wdGray25

colcode = 0

oldColour = Options.DefaultHighlightColorIndex

Options.DefaultHighlightColorIndex = wdGray25

leadDots = " . . . "

title1 = "Proper noun list"

title2 = "Proper noun queries"

convCharsUC = "AAAAAAA.EEEEIIII..OOOOO.OUUUU" & \_

"...aaaaaaa.eeeeiiiio.ooooo.ouuuu......"

convCharsLC = LCase(convCharsUC)

timeStart = Timer

' collect notes text, if any

endText = ""

footText = ""

If FUT.Endnotes.Count > 0 Then

endText = FUT.StoryRanges(wdEndnotesStory).Text

End If

If FUT.Footnotes.Count > 0 Then

footText = FUT.StoryRanges(wdFootnotesStory).Text

End If

' collect text in all the textboxes (if any)

Sh = FUT.Shapes.Count

If Sh > 0 Then

ReDim shText(Sh)

i = 0

For Each shp In FUT.Shapes

If shp.Type <> 24 And shp.Type <> 3 Then

If shp.TextFrame.HasText Then

i = i + 1

shText(i) = shp.TextFrame.TextRange.Text

End If

End If

Next

shCount = i

End If

' Create various documents

Set rng = FUT.Content

Set firstDoc = Documents.Add

Set fnl = firstDoc.Content

Set allText = Documents.Add

Selection.FormattedText = rng.FormattedText

Selection.Collapse wdCollapseEnd

' Add notes + shape text

Selection.TypeText endText & CR & footText & CR

If shCount > 0 Then

For i = 1 To shCount

Selection.TypeText shText(i) & CR

Next i

End If

'Add dummy name

Selection.TypeText Text:="Aaaaa" & CR

Selection.HomeKey Unit:=wdStory

Set rng = allText.Content

rng.Revisions.AcceptAll

DoEvents

StatusBar = spcs & "Preparing copied file - 1"

' Delete struck-through text

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.MatchWildcards = False

.Font.StrikeThrough = True

.Replacement.Text = " "

.Execute Replace:=wdReplaceAll

End With

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "["

.MatchWildcards = False

.Replacement.Text = " "

.Execute Replace:=wdReplaceAll

End With

' Remove strange unicode characters

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[" & ChrW(&HA000) & "-" & ChrW(&HD6FF) & "]{1,}"

.MatchWildcards = True

.Replacement.Text = " "

.Execute Replace:=wdReplaceAll

End With

DoEvents

StatusBar = spcs & "Preparing copied file - 2"

DoEvents

' Cut all and replace as pure text

Set rng = allText.Content

allJustText = allText.Content.Text

allText.Content.Delete

rng.Text = allJustText

DoEvents

StatusBar = spcs & "Preparing copied file - 3"

' Use qqq for apostrophe

Set rng = allText.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "n" & ChrW(8217) & "t"

.MatchWildcards = False

.Replacement.Text = "nqqqt"

.Execute Replace:=wdReplaceAll

End With

' Use qq for apostrophe

With rng.Find

.Text = "O'"

.MatchCase = True

.Replacement.Text = "Oqqq"

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

' Find initial cap words

DoEvents

StatusBar = spcs & "Preparing copied file - 4"

myChopNum = minLengthCheck - 2

If myChop < 1 Then myChop = 1

myChop = Trim(Str(myChopNum))

myFind = "<[A-Z][a-z][a-zA-Z]{" & myChop & ",}"

If includeAcronyms = True Then myFind = \_

"<[A-Z][a-zA-Z][a-zA-Z]{" & myChop & ",}"

Set rng = allText.Range

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = myFind

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.MatchWholeWord = False

.Execute

End With

myPNlist = ""

ignoreWords = ignoreWords & " "

Do While rng.Find.Found = True

myWd = rng.Text

If InStr(ignoreWords, myWd & " ") = 0 Then \_

myPNlist = myPNlist & myWd & CR

rng.Collapse wdCollapseEnd

i = i + 1: If i Mod 100 = 0 Then DoEvents

rng.Find.Execute

DoEvents

Loop

If useQuickSort = True Then

' allText.Content.Text = myPNlist

' wordArray = Split(allText.Content.Text, CR)

' allText.Content.Text = myPNlist

wordArray = Split(myPNlist, CR)

StatusBar = spcs & "Sorting whole file"

' Sort the array alphabetically

Call QuickSort(wordArray, LBound(wordArray), UBound(wordArray))

StatusBar = spcs & "After sorting"

' Clear the content of the document

allText.Content.Delete

' Insert sorted words into the document

For i = LBound(wordArray) To UBound(wordArray)

allText.Content.InsertAfter wordArray(i) & CR

Next i

Else

allText.Content.Text = myPNlist

Set rng = allText.Content

rng.Sort SortOrder:=wdSortOrderAscending, CaseSensitive:=True

End If

' delete initial blank line

If Len(allText.Paragraphs(1)) < 3 Then \_

allText.Paragraphs(1).Range.Delete

If Len(allText.Paragraphs(1)) < 3 Then \_

allText.Paragraphs(1).Range.Delete

allText.Content.InsertAfter Text:=CR

' Create a frequency for each highlighted word

thisWord = ""

myCount = 0

For Each myPara In allText.Paragraphs

nextWord = myPara.Range.Words(1)

If nextWord <> thisWord Then

' This is a new word

If Len(thisWord) > 1 Then

fnl.InsertAfter Text:=thisWord \_

& leadDots & Trim(Str(myCount)) & CR

End If

thisWord = nextWord

myCount = 1

Else

myCount = myCount + 1

End If

i = i + 1: If i Mod 400 = 4 Then DoEvents

Next myPara

allText.Close SaveChanges:=False

' firstDoc.Activate

' Remove blank lines

Set rng = firstDoc.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[^13]{2,}"

.Wrap = wdFindContinue

.Replacement.Text = "^p"

.Forward = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

' Resort case insensitively

Set rng = firstDoc.Content

rng.Sort SortOrder:=wdSortOrderAscending, \_

CaseSensitive:=False

' Delete rubbish from top and bottom of list

Do

Set rng = firstDoc.Paragraphs(1).Range

myLen = Len(rng.Text)

If myLen < 10 Then

rng.Select

Selection.Delete

End If

Loop Until myLen > 9

Do

lastLine = firstDoc.Paragraphs.Count

Set rng = firstDoc.Paragraphs(lastLine).Range

myLen = Len(rng.Text)

If myLen < 10 Then

rng.Select

Selection.Delete

End If

Loop Until Len(rng.Text) >= 2

' Word list now has freq. count.

Do

lastLine = firstDoc.Paragraphs.Count

Set rng = firstDoc.Paragraphs(lastLine).Range

myLen = Len(rng.Text)

If myLen < 10 Then

rng.Select

Selection.Delete

End If

Loop Until Len(rng.Text) >= 2

' Create another copy for doing extra tests

Set rng = firstDoc.Content

Set finalList = Documents.Add

finalList.Range.Text = rng.Text

' Prepare data for other tests

numWords = finalList.Paragraphs.Count

For i = 1 To numWords

aWord = finalList.Paragraphs(i).Range.Words(1)

n = AscW(aWord)

thisChar = ChrW(n)

If n > 129 Then

If n >= 217 Then aWord = Replace(aWord, thisChar, "U")

If n >= 210 Then aWord = Replace(aWord, thisChar, "O")

If n >= 204 Then aWord = Replace(aWord, thisChar, "I")

If n >= 200 Then aWord = Replace(aWord, thisChar, "E")

If n >= 192 Then aWord = Replace(aWord, thisChar, "A")

End If

allWords = allWords & aWord

jmp = 100

If i Mod jmp = 1 Then

PQ = PQ + 1

DoEvents

StatusBar = spcs & \_

"Preparing data for other tests - 1 - " & PQ

DoEvents

End If

Next i

' ...for the vowel test below

DoEvents

StatusBar = spcs & "Preparing data for other tests - 2"

DoEvents

noVowelWords = " " & allWords

noVowelWords = Replace(noVowelWords, " A", "\_1")

noVowelWords = Replace(noVowelWords, " E", "\_2")

noVowelWords = Replace(noVowelWords, " I", "\_3")

noVowelWords = Replace(noVowelWords, " O", "\_4")

noVowelWords = Replace(noVowelWords, " U", "\_5")

noVowelWords = Replace(noVowelWords, " Y", "\_6")

For k = 2 To Len(noVowelWords) - 1

thisChar = Mid(noVowelWords, k, 1)

n = AscW(thisChar)

If n > 191 And n < 221 Then

myNewChar = Mid(convCharsLC, n - 191, 1)

If myNewChar <> "." Then noVowelWords = \_

Replace(noVowelWords, thisChar, myNewChar)

End If

Next k

noVowelWords = Replace(noVowelWords, "a", "")

noVowelWords = Replace(noVowelWords, "e", "")

noVowelWords = Replace(noVowelWords, "i", "")

noVowelWords = Replace(noVowelWords, "o", "")

noVowelWords = Replace(noVowelWords, "u", "")

noVowelWords = Replace(noVowelWords, "y", "")

noVowelWords = Replace(noVowelWords, "A", "")

noVowelWords = Replace(noVowelWords, "E", "")

noVowelWords = Replace(noVowelWords, "I", "")

noVowelWords = Replace(noVowelWords, "O", "")

noVowelWords = Replace(noVowelWords, "U", "")

noVowelWords = Replace(noVowelWords, "Y", "")

noVowelWords = Replace(noVowelWords, "\_1", " A")

noVowelWords = Replace(noVowelWords, "\_2", " E")

noVowelWords = Replace(noVowelWords, "\_3", " I")

noVowelWords = Replace(noVowelWords, "\_4", " O")

noVowelWords = Replace(noVowelWords, "\_5", " U")

noVowelWords = Replace(noVowelWords, "\_6", " Y")

' ...for the similar words test

DoEvents

StatusBar = spcs & "Preparing data for other tests - 3"

DoEvents

similarAllWords = " " & LCase(allWords)

similarChars = Replace(similarChars, " ", "")

sChars = Replace(similarChars, " ", "")

Do

commaPos = InStr(sChars, ",")

charWas = Left(sChars, commaPos - 1)

sChars = Mid(sChars, commaPos + 1)

semicolonPos = InStr(sChars, ";")

charNew = Left(sChars, semicolonPos - 1)

sChars = Mid(sChars, semicolonPos + 1)

similarAllWords = Replace(similarAllWords, charWas, charNew)

Loop Until Len(sChars) < 2

' Changes all the accented characters to non-accented

DoEvents

StatusBar = spcs & "Preparing data for other tests - 4"

DoEvents

sWd = similarAllWords

For k = 1 To Len(sWd) - 1

thisChar = Mid(sWd, k, 1)

n = AscW(thisChar)

myNewChar = "."

If n > 191 And n < 256 Then

myNewChar = Mid(convCharsLC, n - 191, 1)

If myNewChar <> "." Then sWd = Replace(sWd, \_

thisChar, myNewChar)

End If

Next k

similarAllWords = sWd

' Catch words with only the final two letters the same

i = 0

If checkFinalLetters = True Then

For Each myPara In finalList.Paragraphs

gotOne = False

myWord = Trim(myPara.Range.Words(1))

myLen = Len(myWord)

If myLen > 6 Then

myTarget = "^p" & Left(myWord, myLen - 2) & "^$^$ "

myCut = 2

Else

myTarget = "^p" & Left(myWord, myLen - 1) & "^$ "

myCut = 1

End If

Set rng = finalList.Content

rng.Start = myPara.Range.End - 3

rng.Collapse wdCollapseStart

With rng.Find

.Replacement.ClearFormatting

.ClearFormatting

.Text = myTarget

.Replacement.Text = ""

.Forward = True

.MatchCase = True

.MatchWildcards = False

.Wrap = wdFindStop

End With

rng.Find.Execute

Do While rng.Find.Found

gotOne = True

rng.MoveStart 1

rng.End = rng.Start + myLen - myCut

rng.HighlightColorIndex = thisHighlight

rng.Font.Bold = True

rng.Find.Execute

Loop

If gotOne = True Then

Set rng = myPara.Range.Words(1)

rng.End = rng.Start + myLen - myCut

rng.HighlightColorIndex = thisHighlight

rng.Font.Bold = True

End If

i = i + 1

If i Mod 100 = 1 Then

DoEvents

StatusBar = spcs & "Doing test (5) on " & myWord

DoEvents

End If

Next myPara

End If

If doMissingLetter = True Then

' Start of test

doneWords = ""

doneSimilarWords = ""

McList = ""

For i = 1 To finalList.Paragraphs.Count - 1

myWord = finalList.Paragraphs(i).Range.Words(1)

n = AscW(myWord)

thisChar = ChrW(n)

myNewChar = "."

' Changes the capital letter, if a vowel

If n > 191 And n < 221 Then

myNewChar = Mid(convCharsUC, n - 191, 1)

If myNewChar <> "." Then myWord = Replace(myWord, \_

thisChar, myNewChar)

End If

If i Mod 50 = 1 Then

DoEvents

StatusBar = spcs & "Other tests (4) on " & myWord

DoEvents

End If

testWords = Replace(allWords, myWord, "")

captestLetters = Left(myWord, 1)

' Check if word reappears with one letter missing (1)

For k = 2 To Len(myWord) - 1

testWord = " " & Left(myWord, k - 1) & Mid(myWord, k + 1)

wordPos = InStr(allWords, testWord)

If wordPos > 0 Then

lastLetter = Mid(myWord, Len(myWord) - 1, 1)

' but not "s" at the end, unless it's a spelling error

If lastLetter = "s" Then

ignoreIt = (Application.CheckSpelling(myWord, \_

MainDictionary:=myLanguage) = True)

Else

ignoreIt = False

End If

If ignoreIt = False And ignorePlurals = True Then

colcode = (colcode + 1) Mod maxCol

thisCol = myCol(colcode + 1)

' mark the pair

leftBit = Left(allWords, InStr(allWords, testWord) \_

+ Len(testWord) - 1)

j = Len(leftBit) - Len(Replace(leftBit, " ", ""))

Set rng = finalList.Paragraphs(i).Range

rng.HighlightColorIndex = thisCol

rng.Font.Bold = True

rng.Font.Color = wdColorBlue

Set rng = finalList.Paragraphs(j).Range

rng.HighlightColorIndex = thisCol

rng.Font.Bold = True

rng.Font.Color = wdColorBlue

End If

End If

Next k

If Left(myWord, 2) = "Mc" Or Left(myWord, 3) = "Mac" Or \_

Left(myWord, 3) = "Mag" Then

McList = McList & ActiveDocument.Paragraphs(i).Range

End If

Next i

End If

If doSimilarLetters = True Then

doneWords = ""

doneSimilarWords = ""

For i = 1 To finalList.Paragraphs.Count - 1

myWord = finalList.Paragraphs(i).Range.Words(1)

n = AscW(myWord)

thisChar = ChrW(n)

myNewChar = "."

' Changes the capital letter, if a vowel

If n > 191 And n < 221 Then

myNewChar = Mid(convCharsUC, n - 191, 1)

If myNewChar <> "." Then myWord = Replace(myWord, \_

thisChar, myNewChar)

End If

If i Mod 50 = 1 Then

DoEvents

StatusBar = spcs & "Other tests (3) on " & myWord

DoEvents

End If

testWords = Replace(allWords, myWord, "")

captestLetters = Left(myWord, 1)

' check similar spellings: Perutz/Peruts or Chebyshev/Chevychev

similarWord = " " & LCase(myWord)

sChars = similarChars

Do

commaPos = InStr(sChars, ",")

charWas = Left(sChars, commaPos - 1)

sChars = Mid(sChars, commaPos + 1)

semicolonPos = InStr(sChars, ";")

charNew = Left(sChars, semicolonPos - 1)

sChars = Mid(sChars, semicolonPos + 1)

similarWord = Replace(similarWord, charWas, charNew)

Loop Until Len(sChars) < 2

' Changes all the accented characters to non-accented

For k = 1 To Len(myWord) - 1

thisChar = Mid(myWord, k, 1)

n = AscW(thisChar)

If n > 191 And n < 256 Then

myNewChar = Mid(convCharsUC, n - 191, 1)

If myNewChar <> "." Then myWord = Replace(myWord, \_

thisChar, myNewChar)

End If

Next k

similarAllWords = Mid(similarAllWords, Len(similarWord))

theseWords = similarAllWords

If InStr(doneSimilarWords, similarWord) = 0 And \_

InStr(theseWords, similarWord) > 0 Then

colcode = (colcode + 1) Mod maxCol

thisCol = myCol(colcode + 1)

Set rng = finalList.Paragraphs(i).Range

rng.HighlightColorIndex = thisCol

rng.Font.Underline = True

doneSimilarWords = doneSimilarWords & similarWord

' search through all the following words

theseWords = similarAllWords

For j = 1 To numWords - i

spPos = InStr(Trim(theseWords) & " ", " ")

If Left(theseWords, spPos + 1) = similarWord Then

Set rng = finalList.Paragraphs(i + j).Range

rng.HighlightColorIndex = thisCol

rng.Font.Underline = True

End If

theseWords = Mid(theseWords, spPos + 1)

capThisLetter = Mid(theseWords, 2, 1)

If capThisLetter <> LCase(captestLetters) Then Exit For

Next j

End If

Next i

End If

If switchTest = True Then

doneWords = ""

doneSimilarWords = ""

McList = ""

For i = 1 To finalList.Paragraphs.Count - 1

myWord = finalList.Paragraphs(i).Range.Words(1)

n = AscW(myWord)

thisChar = ChrW(n)

myNewChar = "."

' Changes the capital letter, if a vowel

If n > 191 And n < 221 Then

myNewChar = Mid(convCharsUC, n - 191, 1)

If myNewChar <> "." Then myWord = Replace(myWord, \_

thisChar, myNewChar)

End If

If i Mod 50 = 1 Then

DoEvents

StatusBar = spcs & "Other tests (2) on " & myWord

DoEvents

End If

testWords = Replace(allWords, myWord, "")

captestLetters = Left(myWord, 1)

' check for switched chars

wordLen = Len(myWord) - 1

For k = 1 To Len(myWord) - 3

otherWord = Left(myWord, k) & Mid(myWord, k + 2, 1) & \_

Mid(myWord, k + 1, 1) & Mid(myWord, k + 3)

wordPos = InStr(testWords, otherWord)

If wordPos > 0 Then

' Find the position of the matching word

matchWord = Mid(testWords, wordPos, Len(myWord))

leftBit = Left(allWords, InStr(allWords, matchWord) + 1)

j = Len(leftBit) - Len(Replace(leftBit, " ", "")) + 1

finalList.Paragraphs(i).Range.Font.DoubleStrikeThrough \_

= True

finalList.Paragraphs(i).Range.HighlightColorIndex \_

= thisCol

finalList.Paragraphs(j).Range.Font.DoubleStrikeThrough \_

= True

finalList.Paragraphs(j).Range.HighlightColorIndex \_

= thisCol

End If

Next k

Next i

End If

If doVowelTest = True Then

doneWords = ""

doneSimilarWords = ""

McList = ""

For i = 1 To finalList.Paragraphs.Count - 1

myWord = finalList.Paragraphs(i).Range.Words(1)

n = AscW(myWord)

thisChar = ChrW(n)

myNewChar = "."

' Changes the capital letter, if a vowel

If n > 191 And n < 221 Then

myNewChar = Mid(convCharsUC, n - 191, 1)

If myNewChar <> "." Then myWord = Replace(myWord, \_

thisChar, myNewChar)

End If

If i Mod 50 = 1 Then

DoEvents

StatusBar = spcs & "Other tests (1) on " & myWord

DoEvents

End If

testWords = Replace(allWords, myWord, "")

captestLetters = Left(myWord, 1)

' check if there's a word with different vowels

otherWord = " " & Replace(myWord, "a", "")

otherWord = Replace(otherWord, "e", "")

otherWord = Replace(otherWord, "i", "")

otherWord = Replace(otherWord, "o", "")

otherWord = Replace(otherWord, "u", "")

otherWord = Replace(otherWord, "y", "")

' Delete all the accented characters

For k = 3 To Len(otherWord) - 1

thisChar = Mid(otherWord, k, 1)

n = AscW(thisChar)

If InStr("AEIOUY", thisChar) > 0 Then

otherWord = Left(otherWord, k - 1) & "=" & Mid(otherWord, k + 1)

Else

If n > 191 And n < 221 Then

myNewChar = Mid(convCharsUC, n - 191, 1)

If myNewChar <> "." Then

otherWord = Replace(otherWord, thisChar, "=")

End If

End If

End If

Next k

otherWord = Replace(otherWord, "=", "")

' otherWord is now the word under test (vowel-less)

otherWord = Replace(otherWord, ".", "")

noVowelWords = Mid(noVowelWords, Len(otherWord))

If Left(noVowelWords, 1) <> " " Then noVowelWords = \_

" " & noVowelWords

theseWords = noVowelWords

wordPos = InStr(noVowelWords, otherWord)

If InStr(doneWords, otherWord) = 0 And wordPos > 0 Then

colcode = (colcode + 1) Mod maxCol

thisCol = myCol(colcode + 1)

Set rng = finalList.Paragraphs(i).Range

rng.HighlightColorIndex = thisCol

rng.Font.Italic = True

doneWords = doneWords & otherWord

For j = 1 To numWords - i

spPos = InStr(Trim(theseWords) & " ", " ")

firstWord = Left(theseWords, spPos + 1)

theseWords = Mid(theseWords, spPos + 1)

If firstWord = otherWord Then

Set rng = finalList.Paragraphs(i + j).Range

rng.HighlightColorIndex = thisCol

rng.Font.Italic = True

End If

capThisLetter = Mid(theseWords, 2, 1)

If capThisLetter > "" And capThisLetter <> \_

captestLetters Then Exit For

Next j

End If

Next i

End If

finishOff:

finalList.Activate

If Len(McList) > 0 Then

Selection.EndKey Unit:=wdStory

Selection.TypeText CR2 & McList

End If

Selection.HomeKey Unit:=wdStory

If finalList.Content.Words(1) = "Aaaaa " Then \_

finalList.Content.Paragraphs(1).Range.Delete

Selection.TypeText title1 & CR

Selection.Collapse wdCollapseStart

Do

Selection.Expand wdParagraph

If Len(Selection) < 3 Or LCase(Selection) = \_

UCase(Selection) Then Selection.Delete

Loop Until LCase(Selection) <> UCase(Selection)

Selection.HomeKey Unit:=wdStory, Extend:=wdExtend

Selection.Style = finalList.Styles(wdStyleHeading1)

' Restore apostrophes

Set rng = finalList.Range

With rng.Find

.Text = "qqq"

.MatchCase = False

.Replacement.Text = "'"

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

' Find first highlight

Set rng = finalList.Content

With rng.Find

.Text = "Zzzzz"

.Wrap = wdFindContinue

.Replacement.Text = ""

.Forward = True

.MatchWildcards = False

.Execute Replace:=wdReplaceOne

End With

firstDoc.Close SaveChanges:=False

finalList.Activate

' Remove highlighting from second half of words

' that are only case changes of one another

totParas = finalList.Paragraphs.Count

For i = 1 To totParas - 1

a = Trim(finalList.Paragraphs(i).Range.Words(1))

b = Trim(finalList.Paragraphs(i + 1).Range.Words(1))

a = Mid(a, 2)

b = Mid(b, 2)

If LCase(a) = LCase(b) And Len(a) > 2 Then

If (UCase(a) = a And LCase(b) = b) Or (UCase(b) = b And \_

LCase(a) = a) Then

finalList.Paragraphs(i).Range.Words(1).HighlightColorIndex = 0

finalList.Paragraphs(i + 1).Range.Words(1).HighlightColorIndex \_

= 0

End If

End If

If i Mod 50 = 0 Then

DoEvents

StatusBar = spcs & "Final checks: " & totParas - i

DoEvents

End If

Next i

myOnames = ""

Set rng = finalList.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^13O[!a-z]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchSoundsLike = False

.MatchWildcards = True

.Execute

End With

Do While rng.Find.Found = True

rng.Collapse wdCollapseEnd

rng.Expand wdWord

wd = Mid(rng.Text, 3)

rng.Expand wdParagraph

pa = rng.Text

Set rng2 = finalList.Content

With rng2.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^13" & wd

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

If rng2.Find.Found Then

rng2.Collapse wdCollapseEnd

rng2.Expand wdParagraph

pa2 = rng2.Text

myOnames = myOnames & pa2 & pa & vbCr

End If

rng.Collapse wdCollapseEnd

rng.End = rng.End - 2

rng.Find.Execute

Loop

If myOnames > "" Then

Selection.EndKey Unit:=wdStory

Selection.TypeText "Possible O'<something> errors" & vbCr

Selection.MoveUp , 1

Selection.Style = finalList.Styles(wdStyleHeading1)

Selection.EndKey Unit:=wdStory

Selection.TypeText myOnames

Selection.HomeKey Unit:=wdStory

End If

Set rng = finalList.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = myDummy

.Wrap = wdFindContinue

.Replacement.Text = " "

.Forward = True

.MatchCase = False

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^$zzz^$" & leadDots & "1" & vbCr

.Wrap = wdFindContinue

.Replacement.Text = ""

.Forward = True

.MatchCase = False

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

' Clear clipboard

Set rng = finalList.Content

rng.End = 2

rng.Copy

StatusBar = "Creating queries list"

Set rng = finalList.Content

For Each pa In finalList.Paragraphs

pa.Range.Words(1).HighlightColorIndex = wdNoHighlight

Next pa

finalList.Paragraphs(1).Range.HighlightColorIndex = wdNoHighlight

Selection.HomeKey Unit:=wdStory

Set queriesDoc = Documents.Add

Selection.FormattedText = rng.FormattedText

queriesDoc.Paragraphs(1).Range.Delete

Set rng = queriesDoc.Content

rng.Font.StrikeThrough = True

For Each myPara In queriesDoc.Paragraphs

Set ch = myPara.Range.Characters(1)

chCol = ch.HighlightColorIndex

If chCol > 0 Then

myPara.Range.Font.StrikeThrough = False

End If

myLen = Len(myPara.Range.Text)

If myLen > 4 Then

If chCol > 0 Then

myPara.Range.Font.StrikeThrough = False

End If

Set che = myPara.Range.Characters(myLen - 2)

If che.HighlightColorIndex > 0 Then

myPara.Range.Font.StrikeThrough = False

End If

If myPara.Range.Characters(1).Bold = True Then

myPara.Range.Font.StrikeThrough = False

End If

End If

Next myPara

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.StrikeThrough = True

.Wrap = wdFindContinue

.Replacement.Text = "^p"

.Forward = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

DoEvents

End With

Set rng = queriesDoc.Content

rng.Font.StrikeThrough = False

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[^13]{3,}"

.Wrap = wdFindContinue

.Replacement.Text = "^p^p"

.Forward = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

DoEvents

End With

Set rng = queriesDoc.Content

rng.Font.Bold = False

rng.Font.Italic = False

rng.Font.DoubleStrikeThrough = False

rng.Font.Underline = False

rng.Font.Color = wdColorBlack

Selection.HomeKey Unit:=wdStory

Selection.TypeText title2 & CR

Set rng = queriesDoc.Content.Paragraphs(2).Range

If rng.Text = vbCr Then rng.Delete

StatusBar = " "

Options.DefaultHighlightColorIndex = oldColour

lighterColour = wdGray50

Set rng = queriesDoc.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = " [0-9]{1,}"

.Replacement.Font.Italic = False

.Replacement.Text = ""

.Replacement.Highlight = False

.Replacement.Font.Color = wdColorAutomatic

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[a-zA-Z]{1,} "

.Replacement.Text = ""

.Replacement.Font.Italic = False

.Replacement.Font.Color = wdColorAutomatic

.Execute Replace:=wdReplaceAll

DoEvents

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Color = wdColorBlue

.Replacement.Text = ""

.Replacement.Font.Underline = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

Set rng = queriesDoc.Content.Paragraphs(1).Range

rng.Style = queriesDoc.Styles(wdStyleHeading1)

Beep

If keepWholeList = False Then \_

finalList.Close SaveChanges:=False

Application.ScreenUpdating = True

If doingSeveralMacros = False Then

myTime = (Int(10 \* (Timer - timeStart) / 60) / 10)

Beep

If myTime > 0 Then MsgBox myTime & " minutes"

End If

If InStr(FUT.Name, "zzTestFile") > 0 Then

FUT.Activate

Else

queriesDoc.Activate

End If

Exit Sub

ReportIt:

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub

Sub DocAlyse()

' Paul Beverley - Version 18.01.25

' Analyses various aspects of a document

' prompts to count number of tests

cc = 53

myScreenOff = True

Set FUT = ActiveDocument

doingSeveralMacros = (InStr(FUT.Name, "zzTestFile") > 0)

If doingSeveralMacros = False Then

myResponse = MsgBox(" DocAlyse" & vbCr & vbCr & \_

"Analyse this document?", vbQuestion \_

+ vbYesNoCancel, "DocAlyse")

If myResponse <> vbYes Then Exit Sub

End If

If myScreenOff = True Then

Application.ScreenUpdating = False

On Error GoTo ReportIt

End If

ss = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" & \_

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

myTrack = ActiveDocument.TrackRevisions

ActiveDocument.TrackRevisions = False

Set rngOld = ActiveDocument.Content

Documents.Add

Set tempDoc = ActiveDocument

Set rng = ActiveDocument.Content

rng.FormattedText = rngOld.FormattedText

Selection.EndKey Unit:=wdStory

If ActiveDocument.Footnotes.Count > 0 Then

ActiveDocument.StoryRanges(wdFootnotesStory).Copy

Selection.Paste

Selection.Collapse wdCollapseEnd

End If

If ActiveDocument.Endnotes.Count > 0 Then

ActiveDocument.StoryRanges(wdEndnotesStory).Copy

Selection.Paste

Selection.Collapse wdCollapseEnd

End If

Set endTempfile = ActiveDocument.Content

endTempfile.Collapse wdCollapseEnd

' collect text in all the textboxes (if any)

sh = ActiveDocument.Shapes.Count

If sh > 0 Then

For Each shp In ActiveDocument.Shapes

If shp.Type <> 24 And shp.Type <> 3 Then

If shp.TextFrame.HasText Then

shp.TextFrame.TextRange.Copy

endTempfile.Select

Selection.Paste

End If

End If

Next

End If

Selection.HomeKey Unit:=wdStory

Set rng = ActiveDocument.Content

rng.Revisions.AcceptAll

DoEvents

DoEvents

' Use main file for italic 'et al' count...

myTot = ActiveDocument.Range.End

cc = cc - 1

DoEvents

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "<et al>"

.Font.Italic = True

.Replacement.Text = "^&!"

.Wrap = wdFindContinue

.MatchWildcards = True

.MatchWholeWord = False

.MatchSoundsLike = False

.Execute Replace:=wdReplaceAll

italEtAls = ActiveDocument.Range.End - myTot

If italEtAls > 0 Then WordBasic.EditUndo

' ...and superscript degree count

cc = cc - 1

DoEvents

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[oO0]"

.Font.Superscript = True

.Replacement.Text = "vbvb"

.Replacement.Font.Superscript = False

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

funnyDegrees = (ActiveDocument.Range.End - myTot) / 3

.ClearFormatting

.Replacement.ClearFormatting

.Text = " vbvb"

.Replacement.Text = "^&!"

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

funnyDegreesSp = ActiveDocument.Range.End - myTot - funnyDegrees \* 3

If funnyDegreesSp > 0 Then WordBasic.EditUndo

If funnyDegrees > 0 Then WordBasic.EditUndo

DoEvents

Selection.HomeKey Unit:=wdStory

Set rngOld = ActiveDocument.Content

ActiveDocument.TrackRevisions = myTrack

Documents.Add

Set rng = ActiveDocument.Content

rng.FormattedText = rngOld.FormattedText

myEnd = rng.End

Set rng2 = ActiveDocument.Content

rng.Collapse wdCollapseEnd

rng.Text = rng2.Text

Set rng3 = ActiveDocument.Content

rng3.End = myEnd - 1

rng3.Select

Selection.Delete

myRslt = ""

Set rng = ActiveDocument.Content

myTot = ActiveDocument.Range.End

CR = vbCr: CR2 = CR & CR

tr = Chr(9) & "0zczc" & CR: sp = " "

Selection.HomeKey Unit:=wdStory

Set newDoc = ActiveDocument

' Ten or 10

cc = cc - 1

DoEvents

myTot = ActiveDocument.Range.End

rng.Find.ClearFormatting

rng.Find.Replacement.ClearFormatting

rng.Find.Text = "<ten>"

rng.Find.Replacement.Text = "!^&"

rng.Find.MatchWildcards = True

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = " <10>[!,]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "ten" & vbTab & \_

Trim(Str(i)) & CR & "10" & vbTab & Trim(Str(g)) & CR2

' spelt-out lower-case numbers over nine

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<[efnst][efghinorvwx]{2,4}ty"

rng.Find.Execute Replace:=wdReplaceAll

aa = ActiveDocument.Range.End - myTot

If aa > 0 Then WordBasic.EditUndo

rng.Find.Text = "<ten>"

rng.Find.Execute Replace:=wdReplaceAll

ab = ActiveDocument.Range.End - myTot

If ab > 0 Then WordBasic.EditUndo

rng.Find.Text = "<eleven>"

rng.Find.Execute Replace:=wdReplaceAll

ac = ActiveDocument.Range.End - myTot

If ac > 0 Then WordBasic.EditUndo

rng.Find.Text = "<twelve>"

rng.Find.Execute Replace:=wdReplaceAll

ad = ActiveDocument.Range.End - myTot

If ad > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[efnst][efghinuorvwx]{2,4}teen>"

rng.Find.Execute Replace:=wdReplaceAll

ae = ActiveDocument.Range.End - myTot

If ae > 0 Then WordBasic.EditUndo

rng.Find.Text = "<hundred>"

rng.Find.Execute Replace:=wdReplaceAll

af = ActiveDocument.Range.End - myTot

If af > 0 Then WordBasic.EditUndo

If aa + ab + ac + ad + ae + af > 0 Then myRslt = myRslt & \_

"spelt-out numbers (11-999)" & vbTab & \_

Trim(Str(aa + ab + ac + ad + ae + af)) & CR2

' Four-digit numbers

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[!.]<[0-9]{4}>[!,]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

' take off 20xx dates

rng.Find.Text = "[!.]<20[0-9]{2}>[!,]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

' take off 13xx to 19xx dates

rng.Find.Text = "[!.]<1[3-9][0-9]{2}>[!,]"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

i = i - g - k

If i < 0 Then i = 0

' Four figs with comma

rng.Find.Text = "[!.]<[0-9],[0-9]{3}>[!,]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

' Four figs with hard or ordinary space

rng.Find.Text = "[!.]<[0-9][^0160^32][0-9]{3}>[!,]"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If i + g + k > 0 Then

myRslt = myRslt & "Four-digit numbers:" & CR \_

& "nnnn" & vbTab & Trim(Str(i)) & CR \_

& "n,nnn" & vbTab & Trim(Str(g)) & CR \_

& "n nnn" & vbTab & Trim(Str(k)) & CR2

End If

' Dates with 'mid' in front

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "mid [0-9]{4}"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "mid-[0-9]{4}"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "mid[0-9]{4}"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If i + g + k > 0 Then

myRslt = myRslt & "mid 1900(s)" & vbTab \_

& Trim(Str(i)) & CR & "mid-1900(s)" & vbTab & \_

Trim(Str(g)) & CR & "mid1900(s)" & vbTab & \_

Trim(Str(k)) & CR2

End If

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "mid [0-9]{2}[!0-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "mid-[0-9]{2}[!0-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "mid[0-9]{2}[!0-9]"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If i + g + k > 0 Then

myRslt = myRslt & "mid 90(s)" & vbTab \_

& Trim(Str(i)) & CR & "mid-90(s)" & vbTab & \_

Trim(Str(g)) & CR & "mid90(s)" & vbTab & \_

Trim(Str(k)) & CR2

End If

' Serial comma/not serial comma

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[a-zA-Z\-]{1,}, [a-zA-Z\-]{1,}, and "

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

myRslt = myRslt & "serial comma" & vbTab & Trim(Str(i)) & CR

rng.Find.Text = "[a-zA-Z\-]{1,}, [a-zA-Z\-]{1,} and "

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

myRslt = myRslt & "no serial comma" & vbTab & Trim(Str(i)) & CR2

' hard spaces

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "^s"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

' hard hyphens

rng.Find.Text = "^~"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

myRslt = myRslt & "hard spaces" & vbTab & Trim(Str(i)) \_

& CR & "hard hyphens" & vbTab & Trim(Str(g)) & CR2

' Single/double quotes

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = ChrW(8216)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

singleCurl = i

myRslt = myRslt & "curly open single quote" & vbTab & \_

Trim(Str(i)) & CR

rng.Find.Text = ChrW(8220)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

myRslt = myRslt & "curly open double quote" & vbTab & \_

Trim(Str(i)) & CR

rng.Find.Text = Chr(39)

rng.Find.MatchWildcards = True

rng.Find.MatchCase = True

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

myRslt = myRslt & "straight single quote" & vbTab & \_

Trim(Str(i)) & CR

rng.Find.Text = Chr(34)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

myRslt = myRslt & "straight double quote" & vbTab & \_

Trim(Str(i)) & CR2

' etc(.)

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<etc[!.]"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "<etc."

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<etc. [A-Z]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "<etc.^13"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If h + i + g + k > 0 Then myRslt = myRslt & "etc" & \_

vbTab & Trim(Str(h)) & CR & "etc." & vbTab & \_

Trim(Str(i - g - k)) & CR2

' et al(.)

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<et al[!.]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<et al."

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If g + i + italEtAls > 0 Then myRslt = myRslt & "et al." \_

& vbTab & Trim(Str(g)) & CR & "et al (italic, total)" & \_

vbTab & Trim(Str(italEtAls)) & CR & "et al (no dot)" & \_

vbTab & Trim(Str(i)) & CR2

' i.e./ie

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "i.e."

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

myRslt = myRslt

rng.Find.Text = "<ie>"

rng.Find.MatchWildcards = True

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "ie" & vbTab & Trim(Str(g)) & CR \_

& "i.e." & vbTab & Trim(Str(i)) & CR2

' e.g./eg

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "e.g."

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<eg>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "eg" & vbTab & Trim(Str(g)) & CR \_

& "e.g." & vbTab & Trim(Str(i)) & CR2

' Initials with surnames

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<[A-Z]. [A-Z]. [A-Z][a-z]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[A-Z][a-z]{2,}, [A-Z]. [A-Z]. "

rng.Find.Execute Replace:=wdReplaceAll

i2 = ActiveDocument.Range.End - myTot

If i2 > 0 Then WordBasic.EditUndo

aBit = "J. L. B. Matekoni" & vbTab & Trim(Str(i + i2)) & CR

g = i + i2

rng.Find.Text = "<[A-Z].[A-Z]. [A-Z][a-z]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[A-Z][a-z]{2,}, [A-Z].[A-Z]."

rng.Find.Execute Replace:=wdReplaceAll

i2 = ActiveDocument.Range.End - myTot

If i2 > 0 Then WordBasic.EditUndo

aBit = aBit & "J.L.B. Matekoni" & vbTab & Trim(Str(i + i2)) & CR

g = g + i + i2

rng.Find.Text = "<[A-Z] [A-Z] [A-Z][a-z]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[A-Z][a-z]{2,}, [A-Z] [A-Z] "

rng.Find.Execute Replace:=wdReplaceAll

i2 = ActiveDocument.Range.End - myTot

If i2 > 0 Then WordBasic.EditUndo

aBit = aBit & "J L B Matekoni" & vbTab & Trim(Str(i + i2)) & CR

g = g + i + i2

rng.Find.Text = "<[A-Z]{2}> [A-Z][a-z]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[A-Z][a-z]{2,}, [A-Z]{2}"

rng.Find.Execute Replace:=wdReplaceAll

i2 = ActiveDocument.Range.End - myTot

If i2 > 0 Then WordBasic.EditUndo

aBit = aBit & "JLB Matekoni" & vbTab & Trim(Str(i + i2)) & \_

" (Beware! This can be inflated by, e.g. BBC Enterprises.)" & CR2

' Convention for page numbers

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<p. [1-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<pp. [1-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

k = i + g

aBit = "p/pp. 123" & vbTab & Trim(Str(k)) & CR

rng.Find.Text = "<p.[1-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<pp.[1-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

aBit = aBit & "p/pp.123" & vbTab & Trim(Str(i + g)) & CR

k = k + i + g

rng.Find.Text = "<p [1-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<pp [1-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

aBit = aBit & "p/pp 123" & vbTab & Trim(Str(i + g)) & CR

k = k + i + g

rng.Find.Text = "<p[1-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<pp[1-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

aBit = aBit & "p/pp123" & vbTab & Trim(Str(i + g)) & CR2

If k + i + g > 0 Then myRslt = myRslt & aBit

' Convention for ed/eds/edn

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<ed>[!.]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<eds>[!.]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "<edn>[!.]"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "<ed."

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = "<eds."

rng.Find.Execute Replace:=wdReplaceAll

m = ActiveDocument.Range.End - myTot

If m > 0 Then WordBasic.EditUndo

rng.Find.Text = "<edn."

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

If k + m + j + i + g + h > 0 Then myRslt = myRslt \_

& "ed" & vbTab & Trim(Str(i)) & CR & "eds" \_

& vbTab & Trim(Str(g)) & CR & "edn" & vbTab & \_

Trim(Str(h)) & CR & "ed." \_

& vbTab & Trim(Str(k)) & CR & "eds." & vbTab & \_

Trim(Str(m)) & CR & "edn." \_

& vbTab & Trim(Str(j)) & CR2

' Convention for am/pm

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[1-9][ap]m"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

aBit = "2pm" & vbTab & Trim(Str(i)) & CR

rng.Find.Text = "[1-9][ap].m."

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

aBit = aBit & "2p.m." & vbTab & Trim(Str(g)) & CR

rng.Find.Text = "[1-9] [ap]m"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

aBit = aBit & "2 pm" & vbTab & Trim(Str(k)) & CR

rng.Find.Text = "[1-9] [ap].m."

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

aBit = aBit & "2 p.m." & vbTab & Trim(Str(h)) & CR2

If k + i + g + h > 0 Then myRslt = myRslt & aBit

' US/UK spelling

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[bpiv]our[ ,.s]"

rng.Find.Execute Replace:=wdReplaceAll

A = ActiveDocument.Range.End - myTot

If A > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-z]{3,}elling>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-z]{3,}elled>"

rng.Find.Execute Replace:=wdReplaceAll

f = ActiveDocument.Range.End - myTot

If f > 0 Then WordBasic.EditUndo

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[bpiv]or[ ,.s]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "rior[ ,.s]"

rng.Find.Execute Replace:=wdReplaceAll

q = ActiveDocument.Range.End - myTot

If q > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-z]{3,}eling>"

rng.Find.Execute Replace:=wdReplaceAll

v = ActiveDocument.Range.End - myTot

If v > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-z]{3,}eled>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If A + g + f + i + q + v + k > 0 Then myRslt = \_

myRslt & "UK spelling (appx)" & vbTab & \_

Trim(Str(A + g + f)) & CR & \_

"US spelling (appx)" & vbTab & \_

Trim(Str(i - q + v + k)) & CR & \_

"(For a more accurate count, please use UKUScount.)" & CR2

' US/UK punctuation

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[a-zA-Z]['""" & ChrW(8217) & ChrW(8221) & "][,.]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-zA-Z][,.]['""" & ChrW(8217) & ChrW(8221) & "][,.]"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If i + k > 0 Then myRslt = myRslt & \_

"UK punctuation (appx)" & vbTab & \_

Trim(Str(i)) & CR & "US punctuation (appx)" \_

& vbTab & Trim(Str(k)) & CR2

' Initial capital after colon?

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[a-zA-Z]: [A-Z][a-z]"

rng.Find.Execute Replace:=wdReplaceAll

dfgsdfg = ActiveDocument.Range.End

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-zA-Z]: [a-z]"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

If i + j > 0 Then myRslt = myRslt & \_

"Initial capital after colon" & vbTab & \_

Trim(Str(i)) & CR & "Lowercase after colon" \_

& vbTab & Trim(Str(j)) & CR2

' is/iz

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "ise>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "ise[sd]>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "ising>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = "isation"

rng.Find.Execute Replace:=wdReplaceAll

l = ActiveDocument.Range.End - myTot

If l > 0 Then WordBasic.EditUndo

rng.Find.Text = "[armvt]ising"

rng.Find.Execute Replace:=wdReplaceAll

p = ActiveDocument.Range.End - myTot

If p > 0 Then WordBasic.EditUndo

rng.Find.Text = "[arvtw]ise"

rng.Find.Execute Replace:=wdReplaceAll

q = ActiveDocument.Range.End - myTot

If q > 0 Then WordBasic.EditUndo

rng.Find.Text = "ex[eo]rcis[ei]"

rng.Find.Execute Replace:=wdReplaceAll

r = ActiveDocument.Range.End - myTot

If r > 0 Then WordBasic.EditUndo

myRslt = myRslt & "-is- (very appx)" & vbTab & \_

Trim(Str(i + g + k + l - p - q - r)) & CR

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "ize>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "ize[sd]>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "izing>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = "ization"

rng.Find.Execute Replace:=wdReplaceAll

l = ActiveDocument.Range.End - myTot

If l > 0 Then WordBasic.EditUndo

rng.Find.Text = "[Pp]riz[ie]"

rng.Find.Execute Replace:=wdReplaceAll

p = ActiveDocument.Range.End - myTot

If p > 0 Then WordBasic.EditUndo

rng.Find.Text = "[Sse]@iz[ie]"

rng.Find.Execute Replace:=wdReplaceAll

q = ActiveDocument.Range.End - myTot

If q > 0 Then WordBasic.EditUndo

myRslt = myRslt & "-iz- (very appx)" & vbTab \_

& Trim(Str(i + g + k + l - p - q)) & CR & \_

"(For a more accurate count, please use IZIScount.)" \_

& CR2

' data singular/plural

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<data is>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<data has>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "<data was>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[Tt]his data>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

myRslt = myRslt

l = i + g + h + k

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

' If useVoice = True Then speech.Speak cc, SVSFPurgeBeforeSpeak

rng.Find.Text = "<data are>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<data have>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "<data were>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "<[Tt]hese data>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If l + i + h + g + k > 0 Then myRslt = myRslt & \_

"data singular" & \_

vbTab & Trim(Str(l)) & CR & "data plural" & \_

vbTab & Trim(Str(i + g + h + k)) & CR2

' Is "first order" hyphenated?

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[snrt][tdh] order"

rng.Find.Execute Replace:=wdReplaceAll

dfgsdfg = ActiveDocument.Range.End

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[snrt][tdh]-order"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

If i + j > 0 Then myRslt = myRslt & \_

"'xxx order' with space" & vbTab & \_

Trim(Str(i)) & CR & "'xxx-order' with hyphen" \_

& vbTab & Trim(Str(j)) & CR2

' Is "three dimensional" hyphenated?

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[oweunvt ][eonrN1-4i] dimensional"

rng.Find.Execute Replace:=wdReplaceAll

dfgsdfg = ActiveDocument.Range.End

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[oweunvt ][eonrN1-4i]-dimensional"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

If i + j > 0 Then myRslt = myRslt & \_

"'xxx dimensional' with space" & vbTab & \_

Trim(Str(i)) & CR & "'xxx-dimensional' with hyphen" \_

& vbTab & Trim(Str(j)) & CR2

' Types of ellipsis

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = ChrW(8230)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "..."

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

rng.Find.Text = ". . ."

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

If i + j + k > 0 Then

myRslt = myRslt & "Types of ellipsis:" & CR & \_

"proper ellipsis" & vbTab & Trim(Str(i)) & CR \_

& "triple dots" & vbTab & Trim(Str(j)) & CR \_

& "spaced triple dots" & vbTab & Trim(Str(k)) & CR2

End If

' Ellipsis, etc spacing

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

allChars = "/" & ChrW(8211) \_

& ChrW(8212) & "-" & ChrW(8230)

myNames = "Solidus En dash Em dash Hyphen Ellipsis Triple dotsSpaced dots "

For myGo = 0 To 6

sol = Mid(allChars, myGo + 1, 1)

If myGo = 5 Then sol = "..."

If myGo = 6 Then sol = ". . ."

myName = Trim(Mid(myNames, (11 \* myGo) + 1, 11))

rng.Find.Text = sol

rng.Find.Execute Replace:=wdReplaceAll

t = ActiveDocument.Range.End - myTot

If t > 0 Then

WordBasic.EditUndo

rng.Find.Text = " " & sol & " "

rng.Find.Execute Replace:=wdReplaceAll

bth = ActiveDocument.Range.End - myTot

If bth > 0 Then WordBasic.EditUndo

rng.Find.Text = "[! ]" & sol & " "

rng.Find.MatchWildcards = True

rng.Find.Execute Replace:=wdReplaceAll

ftr = ActiveDocument.Range.End - myTot

If ftr > 0 Then WordBasic.EditUndo

rng.Find.Text = " " & sol & "[! ]"

rng.Find.Execute Replace:=wdReplaceAll

bfr = ActiveDocument.Range.End - myTot

If bfr > 0 Then WordBasic.EditUndo

rng.Find.Text = "[! ]" & sol & "[! ]"

rng.Find.Execute Replace:=wdReplaceAll

nthr = ActiveDocument.Range.End - myTot

If nthr > 0 Then WordBasic.EditUndo

myRslt = myRslt & myName & " spacing:" & CR & "space before only" \_

& vbTab & Trim(Str(bfr)) & CR & "space after only" & \_

vbTab & Trim(Str(ftr)) & CR & "spaced both ends" & \_

vbTab & Trim(Str(bth)) & CR

If myGo <> 3 Then

myRslt = myRslt & "not spaced" & vbTab & Trim(Str(nthr)) & CR2

Else

myRslt = myRslt & CR

End If

myRslt = myRslt & CR

End If

Next myGo

' line breaks

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "^l"

rng.Find.MatchWildcards = False

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

' page breaks

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "^m"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

myRslt = myRslt & "line breaks" & vbTab & Trim(Str(i)) \_

& CR & "page breaks" & vbTab & Trim(Str(g)) & CR2

' fig/figure

aBit = ""

rng.Find.Text = "<fig>[!.]"

rng.Find.MatchWildcards = True

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "fig" & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "<Fig>[!.]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "Fig" & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "<fig."

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "fig." & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "<Fig."

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "Fig." & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "<figs>[!.]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "figs" & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "<Figs>[!.]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "Figs" & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "<figs."

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "figs." & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "figure [0-9\(]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "figure" & vbTab & Trim(Str(i)) & CR

End If

rng.Find.Text = "[!.] Figure [0-9\(]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then

WordBasic.EditUndo

aBit = aBit & "Figure" & vbTab & Trim(Str(i)) & CR

End If

If aBit > "" Then myRslt = myRslt & aBit & CR

' Chapter/chapter

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[!.] Chapter [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "chapter [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then

myRslt = myRslt & "Chapter (number)" & vbTab & Trim(Str(i)) & CR \_

& "chapter (number)" & vbTab & Trim(Str(g)) & CR2

End If

' Section/section

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[!.] Section [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "section [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then

myRslt = myRslt & "Section (number)" & vbTab & \_

Trim(Str(i)) & CR & "section (number)" \_

& vbTab & Trim(Str(g)) & CR2

End If

' No./no.

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

' If useVoice = True Then speech.Speak cc, SVSFPurgeBeforeSpeak

rng.Find.Text = " No. [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = " No [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

rng.Find.Text = " no. [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = " No.[0-9]"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = " No[0-9]"

rng.Find.Execute Replace:=wdReplaceAll

l = ActiveDocument.Range.End - myTot

If l > 0 Then WordBasic.EditUndo

rng.Find.Text = " no.[0-9]"

rng.Find.Execute Replace:=wdReplaceAll

m = ActiveDocument.Range.End - myTot

If m > 0 Then WordBasic.EditUndo

If i + j + g + k + l + m > 0 Then

myRslt = myRslt & "No (number)" & vbTab & Trim(Str(i)) \_

& CR & "No. (number)" & vbTab & Trim(Str(j)) & CR \_

& "no. (number)" & vbTab & Trim(Str(g)) & CR

myRslt = myRslt & "No(number)" & vbTab & Trim(Str(k)) \_

& CR & "No.(number)" & vbTab & Trim(Str(l)) & CR \_

& "no.(number)" & vbTab & Trim(Str(m)) & CR2

End If

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = " Vol. [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = " Vol [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

rng.Find.Text = " vol. [0-9]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = " Vol.[0-9]"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = " Vol[0-9]"

rng.Find.Execute Replace:=wdReplaceAll

l = ActiveDocument.Range.End - myTot

If l > 0 Then WordBasic.EditUndo

rng.Find.Text = " vol.[0-9]"

rng.Find.Execute Replace:=wdReplaceAll

m = ActiveDocument.Range.End - myTot

If m > 0 Then WordBasic.EditUndo

If i + j + g + k + l + m > 0 Then

myRslt = myRslt & "Vol (number)" & vbTab & Trim(Str(i)) \_

& CR & "Vol. (number)" & vbTab & Trim(Str(j)) & CR \_

& "vol. (number)" & vbTab & Trim(Str(g)) & CR

myRslt = myRslt & "Vol(number)" & vbTab & Trim(Str(k)) \_

& CR & "Vol.(number)" & vbTab & Trim(Str(l)) & CR \_

& "vol.(number)" & vbTab & Trim(Str(m)) & CR

myRslt = myRslt & CR

End If

' equations

fText = ",<eq [0-9],<eq. [0-9],<eqn [0-9],<Eqn [0-9]," & \_

"eqns [0-9],eqs [0-9],<eq \(,<eq. \(,<Eq. \(" & \_

",<eqn \(,<Eqn \(,eqns \(,eqs \(,Eqs \(,Eqs. \(" & \_

",equation \(,[!.] Equation \(,equations \(" & \_

",[!.] Equations \(,equation [0-9]" & \_

",[!.] Equation [0-9],equations [0-9]" & \_

",[!.] Equations [0-9]"

resText = ",eq,eq.,eqn,Eqn,eqns,eqs,eq (n.n)," & \_

"eq. (n.n),Eq. (n.n),eqn (n.n)," & \_

"Eqn (n.n),eqns (n.n),eqs,Eqs,Eqs.," & \_

"equation (n.n),Equation (n.n)," & \_

"equations (n.n),Equations (n.n),equation," & \_

"Equation,equations,Equations"

myF = Split(fText, ",")

myRes = Split(resText, ",")

aBit = ""

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.MatchWildcards = True

For q = 1 To 23

With rng.Find

.Text = myF(q)

.Replacement.Text = "^&!"

.Execute Replace:=wdReplaceAll

End With

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo: \_

aBit = aBit & myRes(q) & vbTab & Trim(Str(i)) & CR

Next q

If aBit > "" Then myRslt = myRslt & aBit & CR

' units

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[0-9][^32^160][kKcmM][NgAVm]>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9][^32^160][NgAVm]>"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9][kKcmM][NgAVm]>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9][NgAVm]>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

If i + j + g + h > 0 Then

myRslt = myRslt & "spaced units (3 mm)" & vbTab & \_

Trim(Str(i + j)) & CR & "unspaced units (3mm)" \_

& vbTab & Trim(Str(g + h)) & CR2

End If

' Ok, OK, ok, okay

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "<OK>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<Ok>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "<ok>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "<okay>"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

If i + h + g + j > 0 Then myRslt = myRslt & "OK" & \_

vbTab & Trim(Str(i)) & CR \_

& "Ok" & vbTab & Trim(Str(g)) & CR \_

& "ok" & vbTab & Trim(Str(h)) & CR \_

& "okay" & vbTab & Trim(Str(j)) & CR2

' Now go to all lowercase

rng.Case = wdLowerCase

myTot = ActiveDocument.Range.End

' Backward(s), forward(s) etc.

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "[acdlmnostuw][deknort]ward>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[acdlmnostuw][deknort]wards>"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

If i + j > 0 Then myRslt = myRslt & "back/for/toward etc." & \_

vbTab & Trim(Str(i)) & CR \_

& "back/for/towards etc." & vbTab & Trim(Str(j)) & CR2

' amid(st), among(st), while(st)

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "<amid>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "<among>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo: g = g + h

rng.Find.Text = "<while>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo: g = g + h

rng.Find.Text = "<amidst>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "<amongst>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo: i = i + h

rng.Find.Text = "<whilst>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo: i = i + h

If i + g > 0 Then

myRslt = myRslt & "amid/among/while" & vbTab & Trim(Str(g)) & CR

myRslt = myRslt & "amidst/amongst/whilst" & vbTab & Trim(Str(i)) & CR2

End If

' past participle -rnt -elt

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "sp[oi]@lt>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "lea[np]t>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "[l ][be][ua]rnt>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[ds][wpm]elt>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = "sp[oi]@[l]@ed>"

rng.Find.Execute Replace:=wdReplaceAll

p = ActiveDocument.Range.End - myTot

If p > 0 Then WordBasic.EditUndo

rng.Find.Text = "lea[np]ed>"

rng.Find.Execute Replace:=wdReplaceAll

q = ActiveDocument.Range.End - myTot

If q > 0 Then WordBasic.EditUndo

rng.Find.Text = "[l ][be][ua]rned>"

rng.Find.Execute Replace:=wdReplaceAll

r = ActiveDocument.Range.End - myTot

If r > 0 Then WordBasic.EditUndo

rng.Find.Text = "[ds][wpm]elled>"

rng.Find.Execute Replace:=wdReplaceAll

s = ActiveDocument.Range.End - myTot

If g + h + i + k + p + q + r + s > 0 Then myRslt = myRslt & \_

"-rnt -elt" & vbTab & Trim(Str(g + h + i + k)) & CR & \_

"-rned -elled" & vbTab & Trim(Str(p + q + r + s)) & CR2

' percentages

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "[0-9]%"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9][^32^160]%"

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9] per cent>"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9] percent>"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-z]{3,} per cent>"

rng.Find.Execute Replace:=wdReplaceAll

k = ActiveDocument.Range.End - myTot

If k > 0 Then WordBasic.EditUndo

rng.Find.Text = "[a-z]{3,} percent>"

rng.Find.Execute Replace:=wdReplaceAll

m = ActiveDocument.Range.End - myTot

If m > 0 Then WordBasic.EditUndo

If i + j + g + h + k + m > 0 Then

myRslt = myRslt & "unspaced, e.g. 9%" & vbTab & \_

Trim(Str(i)) & CR & "spaced, e.g. 9 %" \_

& vbTab & Trim(Str(j)) & CR & "9 per cent" & vbTab & \_

Trim(Str(g)) & CR & "9 percent" \_

& vbTab & Trim(Str(h)) & CR & "nine per cent" & vbTab & \_

Trim(Str(k)) & CR & "nine percent" \_

& vbTab & Trim(Str(m)) & CR2

End If

' Feet and inches

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

curlyOpt = Options.AutoFormatAsYouTypeReplaceQuotes

Options.AutoFormatAsYouTypeReplaceQuotes = False

rng.Find.Text = "[0-9]'"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9]"""

rng.Find.Execute Replace:=wdReplaceAll

j = ActiveDocument.Range.End - myTot

If j > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9]" & ChrW(8242)

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "[0-9]" & ChrW(8243)

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

Options.AutoFormatAsYouTypeReplaceQuotes = curlyOpt

If i + j + g + h > 0 Then

myRslt = myRslt & "feet (straight) 9'" & vbTab & \_

Trim(Str(i)) & CR & "inches (straight) 9""" \_

& vbTab & Trim(Str(j)) & CR & "single prime 9" & \_

ChrW(8242) & vbTab & Trim(Str(g)) & CR & \_

"double prime 9" & ChrW(8243) & vbTab & \_

Trim(Str(h)) & CR2

End If

' focus(s)

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "focus[ei]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "focuss[ei]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "focus..." & \_

vbTab & Trim(Str(i)) & CR \_

& "focuss..." & vbTab & Trim(Str(g)) & CR2

' benefit(t)

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "benefit[ei]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "benefitt[ei]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "benefit..." & \_

vbTab & Trim(Str(i)) & CR \_

& "benefitt..." & vbTab & Trim(Str(g)) & CR2

' co(-)oper...

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "co-op[ei]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "coop[ei]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "co-oper..." & \_

vbTab & Trim(Str(i)) & CR \_

& "cooper..." & vbTab & Trim(Str(g)) & CR2

' Co-ordin

rng.Find.Text = "co-ord[ei]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "coord[ei]"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

If i + g > 0 Then myRslt = myRslt & "co-ord..." & \_

vbTab & Trim(Str(i)) & CR \_

& "coord..." & vbTab & Trim(Str(g)) & CR2

' Can't, cannot can not

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "can[!a-z]t>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "cannot"

rng.Find.Execute Replace:=wdReplaceAll

g = ActiveDocument.Range.End - myTot

If g > 0 Then WordBasic.EditUndo

rng.Find.Text = "can not"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

If i + h + g > 0 Then myRslt = myRslt & "can't" & \_

vbTab & Trim(Str(i)) & CR \_

& "cannot" & vbTab & Trim(Str(g)) & CR \_

& "can not" & vbTab & Trim(Str(h)) & CR2

' Wasn't, isn't, hasn't

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

myTot = ActiveDocument.Range.End

rng.Find.Text = "[owh ][aie]sn[!a-z]t>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[owh ][aie]s not"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

If i + h > 0 Then myRslt = myRslt & \_

"wasn't, isn't, hasn't" \_

& vbTab & Trim(Str(i)) & CR \_

& "was not, is not, has not" & vbTab & \_

Trim(Str(h)) & CR2

' Don't, won't

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "[dw]on[!a-z]t>"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "[dw][oil]{1,3} not"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

If i + h > 0 Then myRslt = myRslt & \_

"don't, won't" \_

& vbTab & Trim(Str(i)) & CR \_

& "do not, will not" & vbTab & \_

Trim(Str(h)) & CR2

' which/that

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = "which"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = "that"

rng.Find.Execute Replace:=wdReplaceAll

h = ActiveDocument.Range.End - myTot

If h > 0 Then WordBasic.EditUndo

If i + h > 0 Then myRslt = myRslt & \_

"which" \_

& vbTab & Trim(Str(i)) & CR \_

& "that" & vbTab & \_

Trim(Str(h)) & CR2

' Funny characters

cc = cc - 1

DoEvents

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

rng.Find.Text = ChrW(178)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo: \_

myRslt = myRslt & "funny 'squared' character" \_

& vbTab & Trim(Str(i)) & CR2

myDiacritics = ""

For i = 192 To 255

If i <> 215 And i <> 247 Then

myDiacritics = myDiacritics & ChrW(i)

End If

Next i

rng.Find.Text = "[" & myDiacritics & "]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo: \_

myRslt = myRslt & "diacritics" & vbTab & Trim(Str(i)) & CR2

rng.Find.Text = "[" & ChrW(191) & ChrW(161) & ChrW(139) & \_

ChrW(155) & ChrW(171) & ChrW(187) & "]"

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo: myRslt = myRslt & \_

"Continental punctuation" & vbTab & Trim(Str(i)) & CR2

' Ordinary degree symbols

rng.Find.Text = ChrW(176)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = " " & ChrW(176)

rng.Find.Execute Replace:=wdReplaceAll

isp = ActiveDocument.Range.End - myTot

If isp > 0 Then WordBasic.EditUndo

If i > 0 Then myRslt = myRslt & "degree symbols closed" \_

& vbTab & Trim(Str(i - isp)) & CR \_

& "degree symbols spaced" \_

& vbTab & Trim(Str(isp)) & CR2

' Funny degrees

rng.Find.Text = ChrW(186)

rng.Find.Execute Replace:=wdReplaceAll

i = ActiveDocument.Range.End - myTot

If i > 0 Then WordBasic.EditUndo

rng.Find.Text = " " & ChrW(186)

rng.Find.Execute Replace:=wdReplaceAll

isp = ActiveDocument.Range.End - myTot

If isp > 0 Then WordBasic.EditUndo

If i + funnyDegrees > 0 Then

myRslt = myRslt & "funny degrees (0186) closed" \_

& vbTab & Trim(Str(i + funnyDegrees - isp - \_

funnyDegreesSp)) & CR \_

& "funny degrees (0186) spaced" \_

& vbTab & Trim(Str(isp + funnyDegreesSp)) & CR2

End If

appx = ""

If colouredText > 0 Then

If colourOverflow = True Then appx = " (I think)"

myRslt = myRslt & "text in coloured font" \_

& appx & vbTab & Trim(Str(colouredText - 1)) & CR2

End If

If lineBreaks > 0 Then

myRslt = myRslt & "line breaks" \_

& vbTab & Trim(Str(i + lineBreaks)) & CR2

End If

If pageBreaks > 0 Then

myRslt = myRslt & "page breaks" \_

& vbTab & Trim(Str(i + pageBreaks)) & CR2

End If

myRslt = myRslt & CR

Selection.HomeKey Unit:=wdStory

Selection.TypeText Text:="DocAlyse" & vbCr & vbCr

newDoc.Paragraphs(1).Style = ActiveDocument.Styles(wdStyleHeading1)

newDoc.Paragraphs(3).Range.Select

Selection.End = newDoc.Content.End

Selection.TypeText CR & myRslt & CR2

Selection.Font.Bold = True

Set rng = ActiveDocument.Content

rng.ParagraphFormat.TabStops.ClearAll

rng.ParagraphFormat.TabStops.Add Position:=CentimetersToPoints(4.5), \_

Alignment:=wdAlignTabLeft, Leader:=wdTabLeaderSpaces

' Grey out the zero lines

cc = cc - 1

DoEvents

Set rng = ActiveDocument.Content

StatusBar = ss & " " & Trim(Str(cc)) & vbCr

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^13([!^13]@)^t0"

.Wrap = wdFindContinue

.Replacement.Text = "^p\1^t^="

.Replacement.Font.Bold = False

.Replacement.Font.Color = wdColorGray25

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

.Text = "^t^=zczc"

.Wrap = wdFindContinue

.Replacement.Text = ""

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

With Selection.Find

.Text = ""

.Replacement.Text = ""

.MatchWildcards = False

.Execute

End With

Selection.HomeKey Unit:=wdStory

tempDoc.Close SaveChanges:=False

If doingSeveralMacros = False Then

Beep

newDoc.Activate

Else

FUT.Activate

End If

Application.ScreenUpdating = True

Exit Sub

ReportIt:

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub

Sub HyphenAlyse()

' Paul Beverley - Version 05.09.24

' Creates a frequency list of all possible hyphenations

myList = "anti,cross,eigen,hyper,inter,meta,mid,multi," \_

& "non,over,post,pre,pseudo,quasi,semi,sub,super,un"

myList = "anti,cross,eigen,hyper,inter,meta,mid,multi," \_

& "non,over,post,pre,pseudo,quasi,semi,sub,super,un"

includeNumbers = True

deleteTableBorders = True

lighterColour = wdGray25

' lighterColour = wdColor50

Dim myResult As String

myList = "," & myList

myList = Replace(myList, ",,", ",")

pref = Split(myList, ",")

Set FUT = ActiveDocument

doingSeveralMacros = (InStr(FUT.Name, "zzTestFile") > 0)

sp = ChrW(160)

sp = sp & sp & sp

sp = sp & sp & sp

sp = sp & sp & sp

If doingSeveralMacros = False Then

myResponse = MsgBox(" HyphenAlyse" & vbCr & vbCr & \_

"Analyse hyphenated words?", vbQuestion \_

+ vbYesNoCancel, "HyphenAlyse")

If myResponse <> vbYes Then Exit Sub

End If

Dim pr(8000) As String

strttime = Timer

Set rng = ActiveDocument.Content

Documents.Add

Set myDoc = ActiveDocument

Selection.FormattedText = rng.FormattedText

Selection.EndKey Unit:=wdStory

Application.ScreenUpdating = False

On Error GoTo ReportIt

If myDoc.Endnotes.Count > 0 Then Selection.FormattedText = \_

myDoc.StoryRanges(wdEndnotesStory).FormattedText

If myDoc.Footnotes.Count > 0 Then Selection.FormattedText = \_

myDoc.StoryRanges(wdFootnotesStory).FormattedText

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.StrikeThrough = True

.Wrap = wdFindContinue

.Replacement.Text = ""

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

DoEvents

End With

allTheText = myDoc.Content.Text

myDoc.Content.Text = LCase(allTheText)

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ChrW(8217) & "[!a-z]"

.Wrap = wdFindContinue

.Replacement.Text = "!!"

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

DoEvents

End With

If includeNumbers = True Then

schStr = "[a-z0-9]{1,}[-^=][0-9a-z-]{1,}"

Else

schStr = "[a-z]{1,}[-^=][a-z-]{1,}"

End If

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = schStr

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

' Find all hyphenated/dashed word pairs

myPairs = 0

allWords = ","

Do While rng.Find.Found = True

wdPair = Replace(rng.Text, ChrW(8211), "-")

If InStr(allWords, "," & wdPair & ",") = 0 \_

And (UCase(wdPair) <> wdPair) Then

myPairs = myPairs + 1

pr(myPairs) = wdPair

allWords = allWords & wdPair & ","

If myPairs Mod 20 = 0 Then

If doingSeveralMacros = False Then \_

Debug.Print rng.Text & " " & myPairs

StatusBar = sp & rng.Text & " " & myPairs

End If

End If

If Right(wdPair, 1) <> "s" Then

wdPairs = wdPair & "s"

If InStr(allWords, "," & wdPairs & ",") = 0 Then

myPairs = myPairs + 1

pr(myPairs) = wdPairs

allWords = allWords & wdPairs & ","

If myPairs Mod 20 = 0 Then

If doingSeveralMacros = False Then \_

Debug.Print rng.Text, myPairs

StatusBar = sp & rng.Text & " " & myPairs

DoEvents

End If

End If

End If

rng.Find.Execute

Loop

' Collect words with each prefix

For i = 1 To UBound(pref)

hPos = Len(pref(i))

allPreWords = ","

If includeNumbers = True Then

schStr = "<" & pref(i) & "[0-9a-z]{2,}"

Else

schStr = "<" & pref(i) & "[a-z]{2,}"

End If

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = schStr

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

Do While rng.Find.Found = True

wd = rng.Text

If InStr(wd, "-") = 0 Then wd = Left(wd, hPos) \_

& "-" & Mid(wd, hPos + 1)

If InStr(allPreWords, "," & wd & ",") = 0 And \_

InStr(allWords, "," & wd & ",") = 0 Then

myPairs = myPairs + 1

pr(myPairs) = wd

allPreWords = allPreWords & wd & ","

allWords = allWords & wd & ","

If myPairs Mod 20 = 0 Then

If doingSeveralMacros = False Then \_

Debug.Print wd & " " & myPairs

StatusBar = sp & wd & " " & myPairs

DoEvents

End If

End If

rng.Collapse wdCollapseEnd

rng.Find.Execute

Loop

Next i

' Collect word pairs with each prefix, e.g. "mid height"

For i = 1 To UBound(pref)

hPos = Len(pref(i))

If includeNumbers = True Then

schStr = "<" & pref(i) & " [0-9a-z]{2,}"

Else

schStr = "<" & pref(i) & " [a-z]{2,}"

End If

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "<" & pref(i) & " [0-9a-z]{2,}"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

Do While rng.Find.Found = True

wd = rng.Text

If InStr(wd, " ") = 0 Then wd = Left(wd, hPos) \_

& " " & Mid(wd, hPos + 1)

wd = Replace(wd, " ", "-")

If InStr(allPreWords, "," & wd & ",") = 0 And \_

InStr(allWords, "," & wd & ",") = 0 Then

myPairs = myPairs + 1

pr(myPairs) = wd

allPreWords = allPreWords & wd & ","

If myPairs Mod 20 = 0 Then

If doingSeveralMacros = False Then \_

Debug.Print wd & " " & myPairs

StatusBar = sp & wd & " " & myPairs

DoEvents

End If

End If

rng.Collapse wdCollapseEnd

rng.Find.Execute

Loop

Next i

halfTime = Timer

' Count the frequencies

Selection.HomeKey Unit:=wdStory

Selection.TypeText vbCr & vbCr

Selection.HomeKey Unit:=wdStory

ActiveDocument.Paragraphs(1).Style = \_

ActiveDocument.Styles(wdStyleHeading1)

allText = " " & ActiveDocument.Range.Text & " "

' At this point, change all "^p" to "^p "

' all punctuation to " "

chs = " , . ! : ; [ ] { } ( ) / \ + "

chs = chs & ChrW(8220) & " "

chs = chs & ChrW(8221) & " "

chs = chs & ChrW(8201) & " "

chs = chs & ChrW(8222) & " "

chs = chs & ChrW(8217) & " "

chs = chs & ChrW(8216) & " "

chs = chs & ChrW(8212) & " "

chs = chs & ChrW(8722) & " "

chs = chs & vbCr & " "

chs = chs & vbTab & " "

' To force space at start; no space at end

' i.e. one space for each character that

' needs changing to a space

chs = " " & chs & " "

chs = Replace(chs, " ", " ")

chs = Replace(chs, " ", " ")

chs = Left(chs, Len(chs) - 1)

chars = Split(chs, " ")

For i = 1 To UBound(chars)

allText = Replace(allText, chars(i), " ")

Next i

allText = Replace(allText, " ", " ")

cnt = Len(allText)

For i = 1 To myPairs

totFinds = 0

thisFind = ""

Set rng = ActiveDocument.Content

myTot = rng.End

wdHyph = pr(i)

wd = Replace(wdHyph, "-", "")

For j = 1 To 4

Select Case j

Case 1: schWd = wdHyph

Case 2: schWd = Replace(wdHyph, "-", " ")

Case 3: schWd = wd

Case 4: schWd = Replace(wdHyph, "-", ChrW(8211))

End Select

sc = " " & schWd & " "

myCount = Len(Replace(allText, sc, sc & "!")) - cnt

If myCount > 0 Then

totFinds = totFinds + 1

Selection.HomeKey Unit:=wdStory

thisFind = thisFind & schWd & " . ." & \_

Str(myCount) & ":"

Else

thisFind = thisFind & ":"

End If

DoEvents

Next j

If (myPairs - i) Mod 20 = 0 Then

If doingSeveralMacros = False Then \_

Debug.Print "To go: ", myPairs - i

StatusBar = sp & "To go: " & myPairs - i

End If

If Len(thisFind) > 8 Then myResult = myResult & "%" & \_

wd & "%" & thisFind & "!"

Next i

myResult = Replace(myResult, ":!", vbCr)

myResult = Replace(myResult, ":", vbTab)

Selection.WholeStory

Selection.Delete

Set rng = ActiveDocument.Content

rng.InsertAfter myResult

Selection.Sort SortOrder:=wdSortOrderAscending

Selection.HomeKey Unit:=wdStory

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "%\*%"

.Wrap = wdFindContinue

.Replacement.Text = ""

.Forward = True

.MatchCase = False

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

Selection.HomeKey Unit:=wdStory

Selection.TypeText "Hyphenation use"

startTable = Selection.End + 1

ActiveDocument.Paragraphs(1).Style = \_

ActiveDocument.Styles(wdStyleHeading1)

Selection.Start = startTable

Selection.End = ActiveDocument.Range.End

Selection.ConvertToTable Separator:=wdSeparateByTabs

Set TB = ActiveDocument.Tables(1)

For i = 1 To TB.Rows.Count

q = q + 1

If q Mod 20 = 0 Then

Debug.Print "Formatting results: " & q

StatusBar = sp & "Formatting results: " & q

DoEvents

End If

q = 0

Num = 0

For j = 1 To 4

If Len(TB.Cell(i, j).Range.Text) > 2 Then Num = Num + 1

Next j

If Num = 1 Then

For j = 1 To 4

TB.Cell(i, j).Range.Font.ColorIndex = lighterColour

Next j

End If

Next i

Set TB = ActiveDocument.Tables(1)

For i = 1 To TB.Rows.Count

q = q + 1

If q Mod 20 = 0 Then

Debug.Print "Formatting results: " & q

StatusBar = sp & "Formatting results: " & q

DoEvents

End If

q = 0

For j = 1 To 4

hyphPos = 0

txt = TB.Cell(i, j).Range.Text

hyphPos = InStr(txt, "-")

dashPos = InStr(txt, ChrW(8211))

tstText = txt

If hyphPos + dashPos > 0 Then

tstText = "," & Left(txt, hyphPos + dashPos \_

- 1) & ","

If InStr(myList, tstText) > 0 Then

TB.Cell(i, j).Range.Font.ColorIndex = wdBlue

End If

Else

For k = 1 To UBound(pref)

If InStr("," & txt, "," & pref(k)) > 0 Then

TB.Cell(i, j).Range.Font.ColorIndex = wdBlue

End If

Next k

End If

Next j

Next i

For i = 1 To TB.Rows.Count

q = q + 1

If q Mod 20 = 0 Then

Debug.Print "Formatting results: " & q

StatusBar = sp & "Formatting results: " & q

DoEvents

End If

q = 0

s = 0

If Len(TB.Cell(i, 1).Range.Text) > 2 Then s = s + 1

If Len(TB.Cell(i, 3).Range.Text) > 2 Then s = s + 1

If Len(TB.Cell(i, 4).Range.Text) > 2 Then s = s + 1

If Len(TB.Cell(i, 2).Range.Text) > 2 And \_

Len(TB.Cell(i, 4).Range.Text) > 2 Then s = 2

If s > 1 Then

For j = 1 To 4

TB.Cell(i, j).Range.Font.ColorIndex = wdRed

Next j

End If

If InStr(TB.Cell(i, 1).Range.Text, "ly-") > 0 And \_

Len(TB.Cell(i, 2).Range.Text) > 2 Then

For j = 1 To 4

TB.Cell(i, j).Range.Font.ColorIndex = wdRed

Next j

End If

Next i

StatusBar = sp & "Formatting results"

allText = ActiveDocument.Content

For Each myCell In TB.Range.Cells

q = q + 1

If q Mod 20 = 0 Then

Debug.Print "Formatting results: " & q

StatusBar = sp & "Formatting results: " & q

DoEvents

End If

myText = myCell.Range.Text

Set rng = myCell.Range.Duplicate

rng.End = rng.Start + 1

myColour = rng.Font.ColorIndex

i = InStr(myText, " . .")

If myColour = lighterColour And i > 2 Then

myWord = Left(myText, i - 1)

If Right(myWord, 1) = "s" Then

mySingular = Left(myText, i - 2)

If InStr(allText, mySingular & " . .") > 0 Then \_

myCell.Range.Font.Color = wdColorAutomatic

myTest = Replace(mySingular, "-", "")

If InStr(allText, mySingular & " . .") > 0 Then \_

myCell.Range.Font.Color = wdColorAutomatic

myTest = Replace(mySingular, "-", " ")

If InStr(allText, myTest & " . .") > 0 Then \_

myCell.Range.Font.Color = wdColorAutomatic

End If

If InStr(allText, myWord & "s . .") > 0 Then \_

myCell.Range.Font.Color = wdColorAutomatic

myTest = Replace(myWord, "-", "")

If InStr(allText, myTest & "s . .") > 0 Then \_

myCell.Range.Font.Color = wdColorAutomatic

myTest = Replace(myText, "-", " ")

If InStr(allText, myWord & "s . .") > 0 Then \_

myCell.Range.Font.Color = wdColorAutomatic

End If

Next myCell

TB.Style = "Table Grid"

TB.AutoFitBehavior (wdAutoFitContent)

If deleteTableBorders = True Then

TB.Borders(wdBorderTop).LineStyle = wdLineStyleNone

TB.Borders(wdBorderLeft).LineStyle = wdLineStyleNone

TB.Borders(wdBorderBottom).LineStyle = wdLineStyleNone

TB.Borders(wdBorderRight).LineStyle = wdLineStyleNone

TB.Borders(wdBorderHorizontal).LineStyle = wdLineStyleNone

TB.Borders(wdBorderVertical).LineStyle = wdLineStyleNone

End If

Selection.HomeKey Unit:=wdStory

timNow = Timer

Application.ScreenUpdating = True

If doingSeveralMacros = False Then

timGone = timNow - strttime

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

m = Int(timGone / 60)

s = Int(timGone) - m \* 60

timeAll = "Time: " & Trim(Str(m)) & " m " & \_

Trim(Str(s)) & " s"

Selection.HomeKey Unit:=wdStory

numPairs = ActiveDocument.Tables(1).Rows.Count

MsgBox "Items: " & Trim(Str(numPairs)) & vbCr & vbCr \_

& timeAll

Else

FUT.Activate

End If

Exit Sub

ReportIt:

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub

Sub IZtoIS()

' Paul Beverley - Version 01.10.24

' Corrects text to give -is-, -ys- spellings

promptForConfirmation = True

nonoStyles = "DisplayQuote,ReferenceList"

' textColour = wdColorLightBlue

textColour = 0

highlightColour = wdYellow

' highlightColour = 0

bothTCandHighlight = False

closeExceptionsFile = True

' Address where the IZ exceptions file is held

' On Windows, it will need to be something like:

' myZFile = "C:\Documents and Settings\Paul\My Documents\IZ\_words.docx"

' ...but remove the apostrophe in front of myZFile

' On a Mac, it will need to be something like:

' myZFile = "/Users/Paul/My Documents/Macro stuff/IZ\_words.docx"

' ...but remove the apostrophe in front of myZFile' mySFile = "/Users/Paul/My Documents/Macro stuff/IS\_words.docx"

ExceptionZFile = "IZ\_words"

' Start of main program

nonoStyles = "," & nonoStyles & ","

Set mainDoc = ActiveDocument

myTrack = ActiveDocument.TrackRevisions

gottaDoc = False

For Each thisDoc In Application.Documents

thisName = thisDoc.Name

Debug.Print thisDoc

If InStr(thisName, ExceptionZFile) > 0 Then

gottaDoc = True

closeExceptionsFile = False

thisDoc.Activate

Exit For

End If

Next thisDoc

On Error Resume Next

If gottaDoc = False Then

Documents.Open myZFile

If Err.Number = 5174 Then

MsgBox ("Please open the IZ exceptions file")

Err.Clear

Exit Sub

Else

On Error GoTo 0

End If

End If

allWords = "!"

For Each wd In ActiveDocument.Words

thisWord = Trim(wd)

If Len(thisWord) > 2 Then

If Asc(thisWord) > 32 Then allWords = allWords & thisWord & "!"

End If

Next wd

allWords = LCase(allWords)

If closeExceptionsFile = True And gottaDoc = False Then

ActiveDocument.Close SaveChanges:=False

End If

mainDoc.Activate

Selection.HomeKey Unit:=wdStory

If promptForConfirmation = True Then

myResponse = MsgBox("IZ to IS: Edit the text?", vbQuestion + \_

vbYesNoCancel)

Else

myResponse = vbYes

End If

If myResponse = vbCancel Then Exit Sub

If myTrack = True And myResponse = vbYes Then

If bothTCandHighlight = False Then

textColour = 0

highlightColour = 0

End If

End If

If myResponse = vbNo Then ActiveDocument.TrackRevisions = False

totChanges = 0

For hit = 1 To 4

goes = 0

If hit = 1 Then

thisMany = ActiveDocument.Endnotes.Count

If thisMany > 0 Then

Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)

End If

End If

If hit = 2 Then

thisMany = ActiveDocument.Footnotes.Count

If thisMany > 0 Then

Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)

End If

End If

If hit = 3 Then

Set rng = ActiveDocument.Content

Set rng1 = ActiveDocument.Content

thisMany = 1

goes = 1

End If

goes = 1

someText = True

If hit = 4 Then

thisMany = ActiveDocument.Shapes.Count

goes = thisMany

End If

If goes > 0 And thisMany > 0 Then

For myGo = 1 To goes

If hit = 4 Then

Do

someText = False

If ActiveDocument.Shapes(myGo).Type <> 24 And \_

ActiveDocument.Shapes(myGo).Type <> 3 Then

someText = ActiveDocument.Shapes(myGo).TextFrame.HasText

If someText Then

Set rng = ActiveDocument.Shapes(myGo).TextFrame.TextRange

Else

myGo = myGo + 1

End If

End If

DoEvents

Loop Until someText Or myGo > goes

End If

theEnd = rng.End

If someText = True Then

Do

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[iy]z[iea]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.MatchWholeWord = False

.MatchSoundsLike = False

.Execute

End With

If rng.Find.Found = True Then

fnd = rng

opposite = Replace(fnd, "z", "s")

Set rng1 = rng.Duplicate

rng1.Expand wdWord

Do While InStr(ChrW(8217) & "' " & ChrW(160), \_

Right(rng1.Text, 1)) > 0

rng1.MoveEnd , -1

DoEvents

Loop

startWord = rng1.start

endWord = rng1.End

ChangeIt = True

' But don't make the change if...

thisStyle = "," & rng.Style & ","

If InStr(nonoStyles, thisStyle) > 0 Then ChangeIt = False

If rng.Font.StrikeThrough = True Then ChangeIt = False

' if it's not in the list of z's

If InStr(allWords, "!" & LCase(rng1) & "!") = 0 \_

And ChangeIt = True Then

' then definitely change it to an s

If myResponse = vbYes Then

rng.Text = opposite

If ActiveDocument.TrackRevisions = True Then

rng1.End = endWord + 3

Else

rng1.End = endWord

End If

rng1.End = endWord

End If

If highlightColour > 0 Then

rng1.HighlightColorIndex = highlightColour

End If

If textColour > 0 Then

rng1.Font.Color = textColour

End If

totChanges = totChanges + 1

End If

stopNow = False

DoEvents

Else

stopNow = True

End If

rng.start = rng.End + 2

Loop Until stopNow = True

End If

DoEvents

Next myGo

End If

DoEvents

Next hit

ActiveDocument.TrackRevisions = myTrack

If promptForConfirmation = True Then

If myResponse = vbYes Then

MsgBox ("IZ words changed: " & Str(totChanges) & " ")

Else

MsgBox ("IZ words needing to be changed: " & Str(totChanges) & " ")

End If

End If

End Sub

Sub IStoIZ()

' Paul Beverley - Version 01.10.24

' Corrects text to give -iz-, -yz- spellings

promptForConfirmation = False

doExtraWords = False

szExceptions = "analys,reanalys,overanalys,catalys,dialys,"

szExceptions = szExceptions & "electrolys,paralys,hydrolys"

' textColour = wdColorBlue

textColour = 0

highlightColour = wdYellow

highlightColour = 0

analysesColour = 0

analysesColour = wdYellow

bothTCandHighlight = True

nonoStyles = "DisplayQuote,ReferenceList"

closeExceptionsFile = True

doTextBoxes = True

' Address where the IS exceptions file is held

' On Windows, it will need to be something like:

' mySFile = "C:\Documents and Settings\Paul\My Documents\IS\_words.docx"

' ...but remove the apostrophe in front of mySFile

' On a Mac, it will need to be something like:

' mySFile = "/Users/Paul/My Documents/Macro stuff/IS\_words.docx"

' ...but remove the apostrophe in front of mySFile

ExceptionSFile = "IS\_words"

' Start of main program

If doTextBoxes = True Then maxHits = 4 Else maxHits = 3

nonoStyles = "," & nonoStyles & ","

Set mainDoc = ActiveDocument

myTrack = ActiveDocument.TrackRevisions

gottaDoc = False

For Each thisDoc In Application.Documents

thisName = thisDoc.Name

If InStr(thisName, ExceptionSFile) > 0 Then

gottaDoc = True

closeExceptionsFile = False

thisDoc.Activate

Exit For

End If

Next thisDoc

On Error Resume Next

If gottaDoc = False Then

Documents.Open mySFile

If Err.Number = 5174 Then

MsgBox ("Please open the IS exceptions file")

Err.Clear

Exit Sub

Else

On Error GoTo 0

End If

End If

allWords = "!"

For Each wd In ActiveDocument.Words

thisWord = Trim(wd)

If Asc(thisWord) > 32 Then allWords = allWords & thisWord & "!"

Next wd

If closeExceptionsFile = True And gottaDoc = False Then

ActiveDocument.Close SaveChanges:=False

End If

mainDoc.Activate

If promptForConfirmation = True Then

myResponse = MsgBox("IS to IZ: Edit the text?", \_

vbQuestion + vbYesNoCancel)

Else

myResponse = vbYes

End If

If myResponse = vbCancel Then Exit Sub

If myTrack = True And myResponse = vbYes Then

If bothTCandHighlight = False Then

textColour = 0

highlightColour = 0

End If

End If

If myResponse = vbNo Then ActiveDocument.TrackRevisions = False

totChanges = 0

Selection.HomeKey Unit:=wdStory

For hit = 1 To maxHits

goes = 0

If hit = 1 Then

If ActiveDocument.Endnotes.Count > 0 Then

Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)

goes = 1

End If

End If

If hit = 2 Then

If ActiveDocument.Footnotes.Count > 0 Then

Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)

goes = 1

End If

End If

If hit = 3 Then

Set rng = ActiveDocument.Range

goes = 1

End If

If hit = 4 Then

goes = ActiveDocument.Shapes.Count

End If

If goes > 0 Then

For myGo = 1 To goes

someText = True

If hit = 4 Then

Do

someText = False

If ActiveDocument.Shapes(myGo).Type <> 24 And \_

ActiveDocument.Shapes(myGo).Type <> 3 Then

someText = ActiveDocument.Shapes(myGo).TextFrame.HasText

End If

If someText Then

Set rng = ActiveDocument.Shapes(myGo).TextFrame.TextRange

Else

myGo = myGo + 1

End If

DoEvents

Loop Until someText Or myGo > goes

End If

theEnd = rng.End

If someText = True Then

rng.start = 0

rng.End = 0

Do

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[iy]s[iea]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.MatchWholeWord = False

.MatchSoundsLike = False

.Execute

End With

If rng.Find.Found = True Then

fnd = rng

opposite = Replace(fnd, "s", "z")

Set rng1 = rng.Duplicate

rng1.Expand wdWord

Do While InStr(ChrW(8217) & "' " & ChrW(160), \_

Right(rng1.Text, 1)) > 0

rng1.MoveEnd , -1

DoEvents

Loop

startWord = rng1.start

endWord = rng1.End

ChangeIt = True

' But don't make the change if...

thisStyle = "," & rng.Style & ","

If InStr(nonoStyles, thisStyle) > 0 Then ChangeIt = False

If rng.Font.StrikeThrough = True Then ChangeIt = False

' If -is- is near the beginning of the word...

If rng.start - rng1.start < 4 Then

' look for an -is- later in the word

rng.start = rng1.start + 4

rng.End = endWord

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "is[iea]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

opposite = Replace(rng, "s", "z")

If rng.Find.Found = False Or rng.start > endWord \_

Then ChangeIt = False

End If

' Check that it's not in the list of s's

If InStr(allWords, "!" & LCase(rng1) & "!") > 0 \_

Then ChangeIt = False

If InStr(szExceptions, Left(LCase(rng1), 6)) > 0 And \_

rng1.LanguageID = wdEnglishUK Then ChangeIt = False

If LCase(rng1.Text) = "analyses" Then \_

rng1.HighlightColorIndex = analysesColour

If ChangeIt = True Then

' then change it to a z

If myResponse = vbYes Then rng.Text = opposite

If rng1.End < rng.End Then rng1.End = rng.End

If highlightColour > 0 Then

rng1.HighlightColorIndex = highlightColour

End If

If textColour > 0 Then

rng1.Font.Color = textColour

End If

totChanges = totChanges + 1

End If

stopNow = False

Else

stopNow = True

End If

If rng.Text > "" Then

Set rng = rng1.Duplicate

rng.Collapse wdCollapseEnd

End If

i = theEnd - rng.End

If (i Mod 100) = 0 And hit = 3 Then StatusBar = \_

"To go: " & Str(i)

DoEvents

Loop Until stopNow = True

End If

DoEvents

Debug.Print rng.Text

Next myGo

End If

DoEvents

Next hit

ActiveDocument.TrackRevisions = myTrack

If promptForConfirmation = True Then

If myResponse = vbYes Then

MsgBox ("IS words changed: " & Str(totChanges) & " ")

Else

MsgBox ("IS words needing to be changed: " & \_

Str(totChanges) & " ")

End If

End If

End Sub

Sub IZIScount()

' Paul Beverley - Version 12.02.20

' Counts IS/IZ spellings

szExceptions = "analys,reanalys,overanalys,catalys,dialys,"

szExceptions = szExceptions & "electrolys,paralys,hydrolys"

changeZColour = wdTurquoise

changeSColour = wdBrightGreen

nonoStyles = "DisplayQuote,ReferenceList"

exceptionSFile = "IS\_words"

exceptionZFile = "IZ\_words"

closeExceptionsFiles = True

' Address where the IS exceptions file is held

mySFile = "C:\Documents and Settings\Paul\My Documents\IS\_words.docx"

' Address where the IZ exceptions file is held

myZFile = "C:\Documents and Settings\Paul\My Documents\IZ\_words.docx"

Set mainDoc = ActiveDocument

Selection.HomeKey Unit:=wdStory

myResponse = MsgBox("Highlight the IS/IZ words?", \_

vbQuestion + vbYesNoCancel, "ISIZcount")

If myResponse = vbCancel Then Exit Sub

If myResponse = vbYes Then doHighlight = True

gottaSdoc = False

For Each thisDoc In Application.Documents

thisName = thisDoc.Name

If InStr(thisName, exceptionSFile) > 0 Then

gottaSdoc = True

thisDoc.Activate

Exit For

End If

Next thisDoc

On Error Resume Next

If gottaSdoc = False Then

Documents.Open mySFile

If Err.Number = 5174 Then

MsgBox ("Please open the IS exceptions file")

Err.Clear

Exit Sub

Else

On Error GoTo 0

End If

End If

allSWords = "!"

For Each wd In ActiveDocument.Words

thisWord = Trim(wd)

If Asc(thisWord) > 32 Then allSWords = allSWords & thisWord & "!"

Next wd

allSWords = LCase(allSWords)

If closeExceptionsFiles = True And gottaSdoc = False Then

ActiveDocument.Close SaveChanges:=False

End If

gottaZdoc = False

For Each thisDoc In Application.Documents

thisName = thisDoc.Name

If InStr(thisName, exceptionZFile) > 0 Then

gottaZdoc = True

thisDoc.Activate

Exit For

End If

Next thisDoc

If gottaZdoc = False Then

Documents.Open myZFile

If Err.Number = 5174 Then

MsgBox ("Please open the IZ exceptions file")

Err.Clear

Exit Sub

Else

On Error GoTo 0

End If

End If

allZWords = "!"

For Each wd In ActiveDocument.Words

thisWord = Trim(wd)

If Asc(thisWord) > 32 Then allZWords = allZWords & thisWord & "!"

Next wd

allZWords = LCase(allZWords)

If closeExceptionsFiles = True And gottaZdoc = False Then

ActiveDocument.Close SaveChanges:=False

End If

mainDoc.Activate

totZwords = 0

For hit = 1 To 3

If hit = 1 Then

thisMany = ActiveDocument.Endnotes.Count

If thisMany > 0 Then

Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)

Set rng1 = ActiveDocument.StoryRanges(wdEndnotesStory)

End If

End If

If hit = 2 Then

thisMany = ActiveDocument.Footnotes.Count

If thisMany > 0 Then

Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)

Set rng1 = ActiveDocument.StoryRanges(wdFootnotesStory)

End If

End If

If hit = 3 Then

Set rng = ActiveDocument.Content

Set rng1 = ActiveDocument.Content

thisMany = 1

End If

If thisMany > 0 Then

theEnd = rng.End

Do

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[iy]z[iea]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.MatchWholeWord = False

.MatchSoundsLike = False

.Execute

End With

If rng.Find.Found = True Then

fnd = rng

opposite = Replace(fnd, "z", "s")

' Find end of word

rng1.Start = rng.End - 1

rng1.End = rng.End

Do

rng1.End = rng1.End + 1

rng1.Start = rng1.Start + 1

Loop Until UCase(rng1) = LCase(rng1)

wdEnd = rng1.Start

' find start of word

rng1.Start = rng.Start

rng1.End = rng.Start + 1

Do

rng1.End = rng1.End - 1

rng1.Start = rng1.Start - 1

Loop Until UCase(rng1) = LCase(rng1)

' set rng 1 to the whole word

rng1.Start = rng1.End

rng1.End = wdEnd

changeIt = True

' But don't make the change if...

thisStyle = rng.Style

If InStr(nonoStyles, thisStyle) > 0 Then changeIt = False

If rng.Font.StrikeThrough = True Then changeIt = False

' if it's not in the list of z's

If InStr(allZWords, "!" & LCase(rng1) & "!") = 0 And \_

changeIt = True Then

If doHighlight = True Then rng1.HighlightColorIndex = \_

changeZColour

totZwords = totZwords + 1

End If

stopNow = False

Else

stopNow = True

End If

rng.Start = rng.End

i = 2 \* theEnd - rng.End

If (i Mod 100) = 0 And hit = 3 Then StatusBar = "To go: " & Str(i)

Loop Until stopNow = True

End If

Next hit

totSwords = 0

For hit = 1 To 3

If hit = 1 Then

thisMany = ActiveDocument.Endnotes.Count

If thisMany > 0 Then

Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)

Set rng1 = ActiveDocument.StoryRanges(wdEndnotesStory)

End If

End If

If hit = 2 Then

thisMany = ActiveDocument.Footnotes.Count

If thisMany > 0 Then

Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)

Set rng1 = ActiveDocument.StoryRanges(wdFootnotesStory)

End If

End If

If hit = 3 Then

Set rng = ActiveDocument.Content

Set rng1 = ActiveDocument.Content

thisMany = 1

End If

If thisMany > 0 Then

theEnd = rng.End

xcvzxcv = rng.Start

Do

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[iy]s[iea]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

If rng.Find.Found = True Then

fnd = rng

opposite = Replace(fnd, "s", "z")

' Find end of word

rng1.Start = rng.End - 1

rng1.End = rng.End

Do

rng1.End = rng1.End + 1

rng1.Start = rng1.Start + 1

Loop Until UCase(rng1) = LCase(rng1)

wdEnd = rng1.Start

' find start of word

rng1.Start = rng.Start

rng1.End = rng.Start + 1

Do

rng1.End = rng1.End - 1

rng1.Start = rng1.Start - 1

Loop Until UCase(rng1) = LCase(rng1)

' set rng 1 to the whole word

rng1.Start = rng1.End

rng1.End = wdEnd

startWord = rng1.Start

changeIt = True

' But don't make the change if...

thisStyle = rng.Style

If InStr(nonoStyles, thisStyle) > 0 Then changeIt = False

If rng.Font.StrikeThrough = True Then changeIt = False

' If -is- is near the beginning of the word...

If rng.Start - rng1.Start < 4 Then

' look for an -is- later in the word

rng.Start = rng1.Start + 4

rng.End = wdEnd

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "is[iea]"

.Wrap = wdFindStop

.Replacement.Text = ""

.Forward = True

.MatchWildcards = True

.Execute

End With

opposite = Replace(rng, "s", "z")

If rng.Find.Found = False Or rng.Start > wdEnd \_

Then changeIt = False

End If

' Check that it's not in the list of s's

If InStr(allSWords, "!" & LCase(rng1) & "!") > 0 \_

Then changeIt = False

If InStr(szExceptions, Left(LCase(rng1), 6)) > 0 \_

And rng1.LanguageID = wdEnglishUK Then changeIt = False

If changeIt = True Then

If doHighlight = True Then rng1.HighlightColorIndex \_

= changeSColour

totSwords = totSwords + 1

End If

stopNow = False

Else

stopNow = True

End If

rng.Start = wdEnd

rng.End = wdEnd

i = theEnd - rng.End

If (i Mod 100) = 0 And hit = 3 Then StatusBar = "To go: " & Str(i)

Loop Until stopNow = True

End If

Next hit

StatusBar = ""

MsgBox ("IZ words: " & Str(totZwords) & vbCr & vbCr & "IS words: " \_

& Str(totSwords))

End Sub

Sub SpellAlyse()

' Paul Beverley - Version 07.04.23

' Complete spellchecking system

Call MacroNameShow("SpellAlyse")

ignoreNumbers = True

spellingListName = "SpellAlyse"

spellingFreqName = "SpellAlyse frequencies"

ignoreDoc = "zzSwitchList"

myFind = "´a,´e,¨a,¨e,¨o,¨u,ˆo"

myReplace = "á,é,ä,ë,ö,ü,ô"

CR = vbCr

CR2 = CR & CR

sp1 = " "

sp2 = sp1 & sp1

myScreenOff = True

Set FUT = ActiveDocument

doingSeveralMacros = (InStr(FUT.Name, "zzTestFile") > 0)

Dim lossLen As Long

Dim newLen As Long

numParas = FUT.Paragraphs.count

numWords = FUT.Content.ComputeStatistics(wdStatisticWords)

myProfile = numWords / numParas

If myProfile < 1.1 Then GoTo autoCorrect

' List all possible spelling errors

Set rng = ActiveDocument.Content

rng.End = 2

thisLanguage = rng.LanguageID

langText = Languages(rng.LanguageID).NameLocal

langPrompt = "Spellcheck with " & langText & " dictionary. OK?"

If doingSeveralMacros = False Then

myResponse = MsgBox(langPrompt, vbQuestion + vbYesNoCancel, \_

"SpellAlyse")

If myResponse <> vbYes Then Exit Sub

End If

If myScreenOff = True Then

Application.ScreenUpdating = False

On Error GoTo ReportIt

End If

timeStart = Timer

allExcepts = CR

'Collect all words from all word lists

For Each myDoc In Documents

Debug.Print myDoc.Name

pNum = myDoc.Paragraphs.count

myNum = 3

If pNum < 3 Then myNum = pNum

Set rng = myDoc.Paragraphs(myNum).Range

rng.Start = 0

If InStr(LCase(rng.Text), "elist") > 0 Then

For Each myPar In myDoc.Paragraphs

myWord = Trim(myPar.Range.Words(1).Text)

If Len(myWord) > 2 Then

allExcepts = allExcepts & myWord & CR

Debug.Print allExcepts

End If

DoEvents

Next myPar

End If

DoEvents

Next myDoc

Debug.Print allExcepts

' Create a text-only copy in another file

Set rngOld = FUT.Content

Documents.Add

Set erList = ActiveDocument

Set rng = ActiveDocument.Content

rng.LanguageID = thisLanguage

rng.Text = rngOld.Text

numNotes = FUT.Endnotes.count

If numNotes > 0 Then

rng.Collapse wdCollapseEnd

rng.Text = FUT.StoryRanges(wdEndnotesStory).FormattedText

End If

numNotes = FUT.Footnotes.count

If numNotes > 0 Then

rng.Collapse wdCollapseEnd

rng.Text = FUT.StoryRanges(wdFootnotesStory).FormattedText

End If

' copy all the textboxes to the end of the text

shCount = FUT.Shapes.count

If shCount > 0 Then

Selection.EndKey Unit:=wdStory

For j = 1 To shCount

Set shp = FUT.Shapes(j)

If shp.Type <> 24 And shp.Type <> 3 Then

If shp.TextFrame.HasText Then

Set rng = shp.TextFrame.TextRange

If Len(rng.Text) > 1 Then

Selection.Text = rng.Text

Selection.EndKey Unit:=wdStory

End If

End If

End If

DoEvents

Next

End If

' Add a newline for safety

Selection.TypeText CR

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^2"

.Wrap = wdFindContinue

.Replacement.Text = ""

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

.Text = "([!a-zA-Z])['" & ChrW(8216) & "]"

.Replacement.Text = "\1"

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

' Change ligature characters into character pairs

myFind = myFind & "," & ChrW(-1280) & "," & ChrW(-1279) & \_

"," & ChrW(-1278) & "," & ChrW(-1277) & "," & ChrW(-1276) \_

& "," & ChrW(185) & "," & ChrW(178) & "," & ChrW(179)

myReplace = myReplace & ",ff,fi,fl,ffi,ffl, , , , , "

fnd = Split(myFind, ",")

rpl = Split(myReplace, ",")

For i = 0 To UBound(fnd)

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = fnd(i)

.Wrap = wdFindContinue

.Replacement.Text = rpl(i)

.MatchCase = False

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

DoEvents

Next i

Selection.HomeKey Unit:=wdStory

Selection.TypeText Text:="Preparing file for spellchecking" & CR2

ActiveDocument.Paragraphs(1).Style = \_

ActiveDocument.Styles(wdStyleHeading1)

DoEvents

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

If ignoreNumbers = True Then

.Text = "[!a-zA-Z^13'" & ChrW(8217) \_

& ChrW(248) & "-" & ChrW(591) \_

& ChrW(697) & "-" & ChrW(703) \_

& ChrW(&H591) & "-" & ChrW(&H5FF) \_

& ChrW(7680) & "-" & ChrW(7935) & "]"

Else

.Text = "[!a-zA-Z0-9^13'" & ChrW(8217) \_

& ChrW(248) & "-" & ChrW(591) \_

& ChrW(697) & "-" & ChrW(703) \_

& ChrW(&H591) & "-" & ChrW(&H5FF) \_

& ChrW(7680) & "-" & ChrW(7935) & "]"

End If

.Replacement.Text = sp1

.Wrap = wdFindContinue

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

.Text = "([a-zA-Z])['" & ChrW(8217) & "][!a-zA-Z]"

.Replacement.Text = "\1 "

.Execute Replace:=wdReplaceAll

DoEvents

.Text = "^p"

.Replacement.Text = sp1

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

DoEvents

.Text = "^t"

.Execute Replace:=wdReplaceAll

DoEvents

.Text = "^p"

.Execute Replace:=wdReplaceAll

DoEvents

End With

' Create spelling error list

Set rng = ActiveDocument.Content

txt = rng.Text

' Ensure a single space between words

Do

nowLen = Len(txt)

txt = Replace(txt, sp2, sp1)

DoEvents

Loop Until Len(txt) = nowLen

erList1 = ""

erListFreq1 = ""

erList2 = ""

erListFreq2 = ""

If Left(txt, 1) <> sp1 Then txt = sp1 & txt

Do

spPos = InStr(Mid(txt, 2), " ")

wd = Trim(Left(txt, spPos))

spOK = Application.CheckSpelling(wd, MainDictionary:=langText)

spEX = InStr(allExcepts, CR & wd & CR)

nowLen = Len(txt)

txt = Replace(txt, sp1 & wd & sp1, sp1)

txt = Replace(txt, sp1 & wd & sp1, sp1)

lossLen = Len(txt)

If spOK = False And spEX = False And Len(wd) > 2 Then

' record it as a spelling error

numErrs = Int((nowLen - lossLen) / Len(wd))

numText = " . . . " & Trim(Str(numErrs))

If LCase(wd) = wd Then

erList2 = erList2 & wd & CR

erListFreq2 = erListFreq2 & wd & numText & CR

Else

erList1 = erList1 & wd & CR

erListFreq1 = erListFreq1 & wd & numText & CR

End If

StatusBar = wd & numText

Debug.Print wd & numText

End If

DoEvents

Loop Until Len(txt) < 5

Selection.WholeStory

Selection.Delete

Selection.TypeText Replace(erList2, CR2, CR)

Selection.WholeStory

Selection.Sort SortOrder:=wdSortOrderAscending, \_

SortFieldType:=wdSortFieldAlphanumeric

Selection.EndKey Unit:=wdStory

Selection.TypeText CR

listStart = Selection.Start

Selection.TypeText CR & Replace(erList1, CR2, CR)

Selection.Start = listStart

Selection.Sort SortOrder:=wdSortOrderAscending, \_

SortFieldType:=wdSortFieldAlphanumeric

Selection.WholeStory

Selection.LanguageID = thisLanguage

Selection.Style = wdStyleNormal

Selection.Collapse wdCollapseStart

Selection.TypeText spellingListName & vbCr

ActiveDocument.Paragraphs(1).Style = ActiveDocument.Styles(wdStyleHeading1)

Documents.Add

Selection.TypeText Replace(erListFreq2, CR2, CR)

Selection.WholeStory

DoEvents

Selection.Sort SortOrder:=wdSortOrderAscending, \_

SortFieldType:=wdSortFieldAlphanumeric

DoEvents

Selection.EndKey Unit:=wdStory

Selection.TypeText CR

listStart = Selection.Start

Selection.TypeText CR & Replace(erListFreq1, CR2, CR)

Selection.Start = listStart

DoEvents

Selection.Sort SortOrder:=wdSortOrderAscending, \_

SortFieldType:=wdSortFieldAlphanumeric

DoEvents

Selection.WholeStory

Selection.LanguageID = thisLanguage

Selection.Style = wdStyleNormal

Selection.HomeKey Unit:=wdStory

Selection.TypeText spellingFreqName & vbCr

ActiveDocument.Paragraphs(1).Style = ActiveDocument.Styles(wdStyleHeading1)

Selection.Collapse wdCollapseStart

erList.Activate

Selection.HomeKey Unit:=wdStory

Application.ScreenUpdating = True

StatusBar = ""

If doingSeveralMacros = False Then

totTime = Int(10 \* (Timer - timeStart) / 60) / 10

If totTime > 2 Then myResponse = MsgBox((totTime & " minutes"), \_

vbOKOnly, "SpellAlyse")

Beep

Else

FUT.Activate

End If

Exit Sub

autoCorrect:

Application.ScreenUpdating = True

allText = FUT.Content.Text

If InStr(allText, "|") > 0 Then GoTo createListPair

myResponse = MsgBox("Auto-create FRedit list items?", \_

vbQuestion + vbYesNoCancel, "SpellAlyse")

If myResponse <> vbYes Then Exit Sub

Beep

myResponse = MsgBox("Turn track changes on?", \_

vbQuestion + vbYesNoCancel, "SpellAlyse")

If myResponse = vbCancel Then Exit Sub

myRev = (myResponse = vbYes)

If myRev Then ActiveDocument.TrackRevisions = True

' Add auto-created FRedit items

Set rng = ActiveDocument.Content

' (only first character, in case of split language)

rng.End = rng.Start + 1

langName = Languages(rng.LanguageID).NameLocal

i = 0

Do

i = i + 1

Set myPara = FUT.Paragraphs(i)

wd = Replace(myPara.Range.Text, CR, "")

If LCase(wd) = wd And wd <> "" Then

DoEvents

spellOK = Application.CheckSpelling(wd, \_

MainDictionary:=langName)

Set suggList = Application.GetSpellingSuggestions(wd, \_

MainDictionary:=langName)

newWord = ""

DoEvents

If suggList.count > 0 And Not (spellOK) Then

newWord = suggList.Item(1).Name

If myRev = True Then

Set rng = myPara.Range.Duplicate

rng.MoveEnd , -1

myFR = "~<" & wd & ">|" & newWord

rng.Text = myFR

Else

myPara.Range.Text = "~<" & wd & ">|" & newWord & CR

End If

myPara.Range.Select

Selection.Collapse wdCollapseEnd

DoEvents

End If

End If

DoEvents

Loop Until i > 3 And myPara = CR

ActiveDocument.TrackRevisions = False

Exit Sub

createListPair:

' Make Flist + MarkIt list + Elist

Application.ScreenUpdating = True

myResponse = MsgBox("Create exceptions and FRedit lists?", \_

vbQuestion + vbYesNoCancel, "SpellAlyse")

If myResponse <> vbYes Then Exit Sub

Documents.Add

Set eList = ActiveDocument

Selection.TypeText Text:="| Elist" & CR2

eList.Paragraphs(1).Style = eList.Styles(wdStyleHeading1)

Set eRng = ActiveDocument.Content

Documents.Add

Set fList = ActiveDocument

Selection.TypeText Text:="| FRedit" & CR2

fList.Paragraphs(1).Style = fList.Styles(wdStyleHeading1)

Set fRng = ActiveDocument.Content

' FUT is now the spelling error list

For i = 1 To FUT.Paragraphs.count

DoEvents

Set itemRng = FUT.Paragraphs(i).Range

If InStr(itemRng.Text, spellingListName) = 0 \_

And Len(itemRng.Text) > 3 Then

DoEvents

If InStr(itemRng.Text, "|") > 0 Then

fRng.Collapse wdCollapseEnd

fRng.FormattedText = itemRng.FormattedText

Else

myFontCol = itemRng.Font.Color

myHiCol = itemRng.HighlightColorIndex

If myFontCol > 0 Or myHiCol > 0 Then

fRng.Collapse wdCollapseEnd

fRng.Text = "~<" & Replace(itemRng.Text, CR, \_

"") & ">" & "|^&" & CR

fRng.Expand wdParagraph

If myFontCol > 0 Then fRng.Font.Color = myFontCol

If myHiCol > 0 Then fRng.HighlightColorIndex = myHiCol

fRng.Collapse wdCollapseEnd

Else

eRng.Collapse wdCollapseEnd

eRng.Text = itemRng.Text

End If

End If

End If

DoEvents

Next i

Exit Sub

ReportIt:

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub

Sub SpellingErrorHighlighter()

' Paul Beverley - Version 30.10.20

' Highlights all spelling errors

spellingListName = "SpellingErrors"

CR = vbCr

cr2 = CR & CR

sp = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

' Find errors list

Set mainDoc = ActiveDocument

gottaList = False

For Each myDoc In Documents

myFirstLine = myDoc.Paragraphs(1).Range.Text

If InStr(myFirstLine, spellingListName) > 0 Then

myDoc.Activate

gottaList = True

Exit For

End If

Next myDoc

If gottaList = False Then

Beep

MsgBox ("Please load file: """ & spellingListName & """")

Exit Sub

End If

' Create list of words needing highlighting in each colour

Dim myWordHighlightList(16) As String

myCount = 0

For Each par In ActiveDocument.Paragraphs

Set rng = par.Range.Duplicate

If Len(rng) > 2 Then

thisWord = Replace(rng.Text, CR, "")

myCol = rng.HighlightColorIndex

If myCol > 0 And myCol < 17 Then

myWordHighlightList(myCol) = myWordHighlightList(myCol) & \_

thisWord & "\_"

myCount = myCount + 1

StatusBar = sp & sp & sp & myCount

End If

End If

Next par

totCount = myCount

If myCount = 0 Then

Beep

myResponse = MsgBox("Please highlight at least one word in the list!", \_

vbQuestion + vbOK, "SpellingErrorHighlighter")

Exit Sub

End If

mainDoc.Activate

Set rng = ActiveDocument.Content

' To speed up search

Selection.HomeKey Unit:=wdStory

fnNum = ActiveDocument.Footnotes.Count

enNum = ActiveDocument.Endnotes.Count

ActiveDocument.TrackRevisions = False

' For each highlight colour

oldColour = Options.DefaultHighlightColorIndex

For myCol = 1 To 16

If Len(myWordHighlightList(myCol)) > 0 Then

Options.DefaultHighlightColorIndex = myCol

myWds = Split(myWordHighlightList(myCol), "\_")

For Each fWord In myWds

fWord = "<" & fWord & ">"

fWord = Replace(fWord, "(", "\(")

fWord = Replace(fWord, ")", "\)")

fWord = Replace(fWord, "\)>", "\)")

For j = 1 To 3

If j = 1 And fnNum = 0 Then j = 2

If j = 2 And enNum = 0 Then j = 3

Select Case j

Case 1: Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)

Case 2: Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)

Case 3: Set rng = ActiveDocument.Content

End Select

DoEvents

If Len(fWord) > 3 Then

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = fWord

.Replacement.Text = "^&"

.Font.StrikeThrough = False

.Forward = True

.Replacement.Highlight = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "-" & fWord

.Replacement.Text = "^&"

.Font.StrikeThrough = False

.Forward = True

.Replacement.Highlight = False

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

End If

If j = 3 Then

myCount = myCount - 1

StatusBar = sp & sp & sp & "To go: " & myCount

End If

Next j

Next fWord

End If

Next myCol

Options.DefaultHighlightColorIndex = oldColour

Beep

StatusBar = " "

myResponse = MsgBox("All those errors have been highlighted", \_

vbOKOnly, "Spelling Error Highlighter")

End Sub

Sub FRedit()

' Paul Beverley - Version 21.03.25

' Scripted find and replace

' ©2009-2025 Paul Beverley

Set nowDoc = ActiveDocument

promptForSelectedText = True

doFinalBeep = True

showTime = True

useReverseStrikeFeature = False

debugging = False

' debugging = True

funnyCode = "Blank"

' used as the code to mean "Leave the Find/Replace box blank"

isMacro = "DoMacro"

' Used as the code to mean "Do the following macro"

caseCode = ChrW(172)

' The 'bent pipe' character (horizontal line with a bent end)

maxLines = 2000

' the maximum number of F&R lines in your list

myScreenOff = True

oldColour = Options.DefaultHighlightColorIndex

CR = vbCr: CR2 = CR & CR

Set workFile = ActiveDocument

Set rng = workFile.Content

justEditStrikeText = False

editAll = vbNo

If useReverseStrikeFeature = True And \_

rng.Font.StrikeThrough = 9999999 Then

editAll = MsgBox("Edit all text?", \_

vbQuestion + vbYesNoCancel, "FRedit")

If editAll = vbCancel Then Exit Sub

If editAll = vbNo Then

myPrompt = "Edit ONLY strikethrough text (= Yes)" & CR2

myPrompt = myPrompt & "DON'T edit strikethrough text (= No)" & CR2

myPrompt = myPrompt & "Panic! (= Cancel)"

myResponse = MsgBox(myPrompt, vbQuestion + vbYesNoCancel, "FRedit")

If myResponse = vbCancel Then Exit Sub

If myResponse = vbYes Then justEditStrikeText = True

End If

End If

ReDim findText(maxLines) As String, ReplaceText(maxLines) As String

ReDim fHlight(maxLines) As Integer, rHlight(maxLines) As Integer

ReDim fTxtCol(maxLines) As Long, rTxtCol(maxLines) As Long

ReDim fFontSize(maxLines) As Integer, rFontSize(maxLines) As Integer

ReDim styleArray(maxLines, 4) As String, funct(maxLines, 18) As Boolean

timeStart = Timer

For i = 1 To 30

mySpaces = mySpaces & "> "

Next i

myBaseStyle = workFile.Styles(wdStyleNormal)

' In case there's an error

Set thisRng = Selection.Range.Duplicate

' FRedit the selected text only?

Set rng = workFile.Content

If Len(Selection) = Len(rng) Then Selection.HomeKey Unit:=wdStory

Set rngSel = Selection.Range.Duplicate

If Selection.End <> Selection.start Then

If promptForSelectedText = True Then

myResponse = MsgBox("Work on selected text only?", \_

vbQuestion + vbYesNoCancel, "FRedit")

If myResponse = vbCancel Then Exit Sub

Else

myResponse = vbYes

End If

wasSelectedText = (myResponse = vbYes)

Else

wasSelectedText = False

End If

If debugging = False Then On Error GoTo ReportIt

If myScreenOff = True Then Application.ScreenUpdating = False

' Assume cursor is in the file to be edited

Set workFile = ActiveDocument

myTrack = workFile.TrackRevisions

' Find the FRedit list file

gottaList = 0

For Each myDoc In Application.Documents

DoEvents

pNum = myDoc.Paragraphs.count

myNum = 5

If pNum < 5 Then myNum = pNum

Set rng = myDoc.Paragraphs(myNum).Range

rng.start = 0

myTest = Replace(LCase(rng), " ", "")

If InStr(myTest, ChrW(124) & "fredit") Then

gottaList = gottaList + 1

If gottaList = 1 Then

myDoc.Activate

Set theList = myDoc

End If

End If

' Debug.Print myDoc.Name

Next myDoc

myWarning = "Please ensure that your FRedit list starts with: | FRedit" \_

& CR & CR2 & "Then place the cursor in the text to be edited, and rerun FRedit."

If gottaList = 0 Then

Beep

myResponse = MsgBox("Can't find a FRedit LIST." & CR2 & \_

myWarning, vbExclamation + vbOKOnly, "FRedit")

Exit Sub

End If

If gottaList > 1 Then

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

myResponse = MsgBox("There are TWO or more FRedit LISTs open!" & CR2 & \_

"Is this the list you want to use?", vbQuestion + vbYesNo, "FRedit")

If myResponse <> vbYes Then

myResponse = MsgBox("Please close the list(s) you don't want to use," & \_

CR & "and then rerun FRedit", vbExclamation + vbOKOnly, "FRedit")

Exit Sub

End If

End If

' Debug.Print workFile.Name, theList.Name

If workFile = theList Then

Beep

myResponse = MsgBox("The CURSOR seems to be in the FRedit LIST." & CR2 & \_

"Place the cursor in the TEXT TO BE EDITED, and rerun FRedit.", \_

vbExclamation + vbOKOnly, "FRedit")

Exit Sub

End If

Set rng = theList.Content

' Check final line in the list is a paragraph marker

If Right(rng, 2) <> CR2 Then rng.InsertAfter Text:=CR

' Check for rogue manual line breaks in the list

mlb = InStr(rng.Text, Chr(11))

If mlb > 0 Then

Selection.start = mlb - 1

Selection.MoveLeft , 1

Selection.End = mlb + 1

Beep

MsgBox "Beware: FRedit list contains manual line breaks!"

Application.Run macroName:="ShowFormatting"

MsgBox "I've turned Show Formatting on, so you can see."

Application.ScreenUpdating = True

Exit Sub

End If

' Create the list of F&Rs

Selection.HomeKey Unit:=wdStory

FRitem = 0

fNotes = False: eNotes = False: Etext = True: BoxText = False

' FRitem is used to count the actual lines that are F&R commands

lastLine = theList.Paragraphs.count

For i = 1 To lastLine

' Look through the list for things that aren't actual F&Rs

Do

Do

Set rng = theList.Paragraphs(i).Range

StatusBar = mySpaces & "Reading line: " & Str(FRitem)

DoEvents

rng.End = rng.End - 1

allLine = rng

i = i + 1

' Keep going until you find a non-blank line

Loop Until Len(allLine) > 0 Or i > lastLine

' Check if it's a comment line, i.e. starting with a pad

firstChar = Left(allLine, 1)

If firstChar = "|" Then

If InStr(allLine, "rack =") > 0 Then

' Check for | Track command

Select Case LCase(Right(allLine, 3))

Case " no"

If myTrack Then

MsgBox "Please switch track changes off!", \_

vbOKOnly + vbExclamation, "FRedit"

workFile.Activate

Application.ScreenUpdating = True

Exit Sub

End If

Case "yes"

If myTrack = False Then

MsgBox "Please switch track changes on!", \_

vbOKOnly + vbExclamation, "FRedit"

workFile.Activate

Application.ScreenUpdating = True

Exit Sub

End If

Case Else

myError = 4: GoTo myErrorReport

End Select

End If

If InStr(allLine, "ootnotes =") > 0 Then

' Check for | Footnote command

If InStr(LCase(allLine), "yes") > 0 Then

fNotes = True

Else

fNotes = False

End If

End If

If InStr(allLine, "ndnotes =") > 0 Then

' Check for | Endnote command

If InStr(LCase(allLine), "yes") > 0 Then

eNotes = True

Else

eNotes = False

End If

End If

If InStr(allLine, "ext =") > 0 Then

' Check for | Text command

If InStr(LCase(allLine), "yes") > 0 Then

Etext = True

Else

Etext = False

End If

End If

If InStr(allLine, "box =") > 0 Or InStr(allLine, \_

"boxes =") > 0 Then

' Check for | Textboxes command

If InStr(LCase(allLine), "yes") > 0 Then

BoxText = True

Else

BoxText = False

End If

End If

End If

Loop Until firstChar <> "|" Or i > lastLine

i = i - 1

' If you find hashes, stop looking for F&R lines

If Left(allLine, 1) = "#" Then Exit For

' Debug.Print allLine

' It's an F&R line, so check for highlighting

If Len(allLine) > 1 Then

lineStart = rng.start

lineEnd = rng.End

' Has it got a vertical bar in it?

padPosition = InStr(allLine, "|")

' If not, it's a two-line F&R

If padPosition = 0 Then

' We've got two lines

fText = rng

' Clip off MatchCase and Wildcard indicators

mchWild = False

mchCase = True

If Left(fText, 1) = caseCode Then

fText = Right(fText, Len(fText) - 1)

mchCase = False

rng.start = rng.start + 1

End If

If Left(fText, 1) = "~" Then

fText = Right(fText, Len(fText) - 1)

mchWild = True

rng.start = rng.start + 1

End If

' What style is the Find in?

fStyle = rng.Style

If fStyle = myBaseStyle Then fStyle = ""

rng.End = lineStart + 1

' Check format & type colour of first char of Find

fItalic = rng.Italic

fBold = rng.Bold

fSuper = rng.Font.Superscript

fSub = rng.Font.Subscript

fUline = rng.Underline

fSmall = rng.Font.SmallCaps

fAllcaps = rng.Font.AllCaps

fDstrike = rng.Font.DoubleStrikeThrough

fFont = rng.Font.Name

fSize = rng.Font.Size

fHiColour = rng.HighlightColorIndex

fTxtColour = rng.Font.Color

i = i + 1

Set rng = theList.Paragraphs(i).Range

rng.End = rng.End - 1

lineEnd = rng.End

rText = rng

padPosition = InStr(rText, "|")

If padPosition > 0 Or Len(rng) = 0 Then myError = 2: GoTo myErrorReport

' What style is the Replace in?

rstyle = rng.Style

If rstyle = myBaseStyle Then rstyle = ""

rng.End = rng.start + 1

' Check format & type colour of first char of Replace

rItalic = rng.Italic

rBold = rng.Bold

rSuper = rng.Font.Superscript

rSub = rng.Font.Subscript

rUline = rng.Underline

rSmall = rng.Font.SmallCaps

rAllcaps = rng.Font.AllCaps

rDstrike = rng.Font.DoubleStrikeThrough

rFont = rng.Font.Name

rSize = rng.Font.Size

rHiColour = rng.HighlightColorIndex

rTxtColour = rng.Font.Color

Else

' It's all on one line, so no style change

' unless the style is different from Normal

' What style is the Find in?

fStyle = ""

rstyle = rng.Style

If rstyle = myBaseStyle Then rstyle = ""

' Chop up the line into F and R

fText = Left(allLine, padPosition - 1)

rText = Right(allLine, Len(allLine) - padPosition)

' Clip off MatchCase and Wildcard indicators

mchWild = False

mchCase = True

If Left(fText, 1) = caseCode Then

fText = Right(fText, Len(fText) - 1)

mchCase = False

rng.start = rng.start + 1

End If

If Left(fText, 1) = "~" Then

fText = Right(fText, Len(fText) - 1)

mchWild = True

rng.start = rng.start + 1

End If

rng.End = lineStart + 1

' Check format & type colour of first char of Find

If rng.Text = " " Then

rng.start = rng.End

rng.End = rng.End + 1

End If

fItalic = rng.Italic

fBold = rng.Bold

fSuper = rng.Font.Superscript

fSub = rng.Font.Subscript

fUline = rng.Underline

fSmall = rng.Font.SmallCaps

fAllcaps = rng.Font.AllCaps

fDstrike = rng.Font.DoubleStrikeThrough

fFont = rng.Font.Name

fSize = rng.Font.Size

fHiColour = rng.HighlightColorIndex

fTxtColour = rng.Font.Color

rng.End = lineStart + padPosition + 1

rng.start = lineStart + padPosition

' Check format & type colour of first char of Replace

rItalic = rng.Italic

rBold = rng.Bold

rSuper = rng.Font.Superscript

rSub = rng.Font.Subscript

rUline = rng.Underline

rSmall = rng.Font.SmallCaps

rAllcaps = rng.Font.AllCaps

rDstrike = rng.Font.DoubleStrikeThrough

rFont = rng.Font.Name

rSize = rng.Font.Size

rHiColour = rng.HighlightColorIndex

rTxtColour = rng.Font.Color

trackit = Not (rng.Font.StrikeThrough)

End If

FRitem = FRitem + 1

' Save all the F&R info in arrays

findText(FRitem) = fText

ReplaceText(FRitem) = rText

styleArray(FRitem, 1) = fStyle

styleArray(FRitem, 2) = rstyle

styleArray(FRitem, 3) = fFont

styleArray(FRitem, 4) = rFont

fHlight(FRitem) = fHiColour

rHlight(FRitem) = rHiColour

fTxtCol(FRitem) = fTxtColour

rTxtCol(FRitem) = rTxtColour

fFontSize(FRitem) = fSize

rFontSize(FRitem) = rSize

funct(FRitem, 1) = mchWild

funct(FRitem, 2) = mchCase

funct(FRitem, 3) = fBold

funct(FRitem, 4) = rBold

funct(FRitem, 5) = fItalic

funct(FRitem, 6) = rItalic

funct(FRitem, 7) = fSuper

funct(FRitem, 8) = rSuper

funct(FRitem, 9) = fSub

funct(FRitem, 10) = rSub

funct(FRitem, 11) = fUline

funct(FRitem, 12) = rUline

funct(FRitem, 13) = fSmall

funct(FRitem, 14) = rSmall

funct(FRitem, 15) = fAllcaps

funct(FRitem, 16) = rAllcaps

funct(FRitem, 17) = trackit

funct(FRitem, 18) = fDstrike

' ^p is not allowed in wildcard searches!

If mchWild And InStr(fText, "^" & "p") > 0 Then

myError = 5: GoTo myErrorReport

End If

If InStr(fText, "^" & "{") > 0 Then

myError = 95: GoTo myErrorReport

End If

' You can't do case insensitive AND wildcard

If mchWild And mchCase = False Then

myError = 6: GoTo myErrorReport

End If

End If

Next i

lastItem = FRitem

' Check the Normal font

normalSize = theList.Styles(myBaseStyle).Font.Size

normalFont = theList.Styles(myBaseStyle).Font.Name

workFile.Activate

' Remember if TC is on or off

trackNow = workFile.TrackRevisions

' Remember current cursor position and move cursor to top, for speed

Set thisRng = Selection.Range.Duplicate

Selection.HomeKey Unit:=wdStory

' Define the ranges

' Get the data out of the arrays

For FRitem = 1 To lastItem

fText = findText(FRitem)

rText = ReplaceText(FRitem)

fStyle = styleArray(FRitem, 1)

rstyle = styleArray(FRitem, 2)

fFont = styleArray(FRitem, 3)

rFont = styleArray(FRitem, 4)

fHiColour = fHlight(FRitem)

rHiColour = rHlight(FRitem)

fSize = fFontSize(FRitem)

rSize = rFontSize(FRitem)

fTxtColour = fTxtCol(FRitem)

rTxtColour = rTxtCol(FRitem)

mchWild = funct(FRitem, 1)

mchCase = funct(FRitem, 2)

fBold = funct(FRitem, 3)

rBold = funct(FRitem, 4)

fItalic = funct(FRitem, 5)

rItalic = funct(FRitem, 6)

fSuper = funct(FRitem, 7)

rSuper = funct(FRitem, 8)

fSub = funct(FRitem, 9)

rSub = funct(FRitem, 10)

fUline = funct(FRitem, 11)

rUline = funct(FRitem, 12)

fSmall = funct(FRitem, 13)

rSmall = funct(FRitem, 14)

fAllcaps = funct(FRitem, 15)

rAllcaps = funct(FRitem, 16)

trackit = funct(FRitem, 17)

fDstrike = funct(FRitem, 18)

If trackNow = True Then workFile.TrackRevisions = trackit

If fText = isMacro Then

Application.Run macroName:=rText

Else

' funnyCode means fText should be blank

If InStr(fText, funnyCode) > 0 Then fText = ""

BlankIt = False

If InStr(rText, funnyCode) > 0 Then rText = "": BlankIt = True

If fText = "<Symbol>" Then

fText = "": BlankIt = True

fFont = "Symbol"

rFont = normalFont

End If

' Replace hex code strings with codes

codePos = InStr(fText, "<&H")

Do While codePos > 0

codeLen = InStr(fText, ">") - codePos

uText = Mid(fText, codePos, codeLen)

uCode = Val(Right(uText, codeLen - 1))

uChar = ChrW(uCode)

fText = Replace(fText, uText & ">", uChar)

codePos = InStr(fText, "<&H")

Loop

codePos = InStr(rText, "<&H")

Do While codePos > 0

codeLen = InStr(rText, ">") - codePos

uText = Mid(rText, codePos, codeLen)

uCode = Val(Right(uText, codeLen - 1))

uChar = ChrW(uCode)

rText = Replace(rText, uText & ">", uChar)

codePos = InStr(rText, "<&H")

Loop

For hit = 1 To 4

If hit = 1 Then

If Not (fNotes = True And workFile.Footnotes.count \_

> 0) Then hit = 2

End If

If hit = 2 Then

If Not (eNotes = True And workFile.Endnotes.count \_

> 0) Then hit = 3

End If

If hit = 3 Then

If Etext = False Then hit = 4

End If

goes = 1

If hit = 4 Then

If BoxText = True Then

goes = workFile.Shapes.count

Else

hit = 5

End If

End If

If hit < 5 Then

For myGo = 1 To goes

If hit = 1 Then Set rng = \_

workFile.StoryRanges(wdFootnotesStory)

If hit = 2 Then Set rng = \_

workFile.StoryRanges(wdEndnotesStory)

If hit = 3 Then

If wasSelectedText Then

Set rng = rngSel.Duplicate

Else

Set rng = workFile.Content

End If

End If

someText = True

If hit = 4 Then

Do

someText = False

If workFile.Shapes(myGo).Type <> 24 \_

And workFile.Shapes(myGo).Type <> 3 Then

someText = workFile.Shapes(myGo).TextFrame.HasText

End If

If someText Then

Set rng = workFile.Shapes(myGo).TextFrame.TextRange

Else

myGo = myGo + 1

End If

Loop Until someText Or myGo > goes

End If

If someText = True Then

' Now do the F&R with the appropriate conditions set

Options.DefaultHighlightColorIndex = rHiColour

If (rHiColour <> fHiColour) And (fHiColour <> 0) Then

' But first emboss all text in fHiColour

Set rngNow = rng.Duplicate

For Each myPar In rngNow.Paragraphs

If myPar.Range.HighlightColorIndex > 9999 Then

For Each wd In myPar.Range.Words

If wd.HighlightColorIndex > 9999 Then

For Each ch In wd.Characters

If ch.HighlightColorIndex = fHiColour Then

ch.Font.Emboss = True

End If

Next ch

Else

If wd.HighlightColorIndex = fHiColour Then

wd.Font.Emboss = True

End If

End If

Next wd

Else

If myPar.Range.HighlightColorIndex = fHiColour Then

If Len(myPar.Range.Text) > 1 Then myPar.Range.Font.Emboss = True

End If

End If

DoEvents

Next myPar

End If

StatusBar = mySpaces & "F&Ring line: " & Str(FRitem) & \_

" of " & Str(lastItem) & " > > > " & fText & \_

"|" & rText

Debug.Print Str(FRitem) & " of " & Str(lastItem) & \_

" > " & fText & " | " & rText

' Now do the F&R

Set rngNow = rng.Duplicate

With rngNow.Find

.ClearFormatting

.Replacement.ClearFormatting

.Format = False

If wasSelectedText Then

.Wrap = False

Else

.Wrap = wdFindContinue

End If

.Text = fText

.Replacement.Text = rText

.MatchWildcards = mchWild

.MatchCase = mchCase

DoEvents

If fStyle > "" Or rstyle > "" Then

If rstyle = "" Then rstyle = myBaseStyle

If fStyle > "" Then .Style = fStyle

If rstyle > "" Then .Replacement.Style = rstyle

Else

' N.B. If changing styles, don't try to

' change bold, italic, etc, etc.

If fBold <> rBold Then

.Font.Bold = fBold

.Replacement.Font.Bold = rBold

End If

If rBold Then .Replacement.Font.Bold = True

If fItalic <> rItalic Then

.Font.Italic = fItalic

.Replacement.Font.Italic = rItalic

End If

If rItalic Then .Replacement.Font.Italic = True

If fSuper <> rSuper Then

.Font.Superscript = fSuper

.Replacement.Font.Superscript = rSuper

End If

If rSuper Then .Replacement.Font.Superscript = True

If fSub <> rSub Then

.Font.Subscript = fSub

.Replacement.Font.Subscript = rSub

End If

If rSub Then .Replacement.Font.Subscript = True

If fUline <> rUline Then

.Font.Underline = fUline

.Replacement.Font.Underline = rUline

End If

If rUline Then .Replacement.Font.Underline = True

If fSmall <> rSmall Then

.Font.SmallCaps = fSmall

.Replacement.Font.SmallCaps = rSmall

End If

If rSmall Then .Replacement.Font.SmallCaps = True

If fAllcaps <> rAllcaps Then

.MatchCase = fAllcaps

.Font.AllCaps = rAllcaps

.Replacement.Font.AllCaps = False

End If

If rAllcaps Then

.MatchCase = False

.Replacement.Font.AllCaps = True

End If

If rDstrike Then .Replacement.Font.StrikeThrough = True

If fStyle = rstyle Then

If fFont = rFont And fFont <> normalFont Then

.Replacement.Font.Name = rFont

End If

If fFont <> rFont Then

.Font.Name = fFont

.Replacement.Font.Name = rFont

End If

If fSize = rSize And fSize <> normalSize Then

.Replacement.Font.Size = rSize

End If

If fSize <> rSize Then

.Font.Size = fSize

.Replacement.Font.Size = rSize

End If

End If

End If

If rHiColour <> fHiColour And fHiColour <> 0 And rText > "" And \_

fHiColour > 0 Then .Font.Emboss = True

If rText > "" And (rHiColour > 0 Or fHiColour > 0) Then

.Replacement.Highlight = True

If rHiColour = 0 Then .Replacement.Highlight = False

End If

If rText > "" Then

If rTxtColour = fTxtColour Then

If fTxtColour > 0 Then \_

.Replacement.Font.Color = fTxtColour

Else

If rTxtColour > 0 Or fTxtColour > 0 Then

.Font.Color = fTxtColour

If BlankIt = False Then .Replacement.Font.Color = \_

rTxtColour

End If

End If

End If

If justEditStrikeText = True Then

.Font.StrikeThrough = True

Else

If editAll = vbNo Then .Font.StrikeThrough = False

End If

.Font.DoubleStrikeThrough = False

.Font.Hidden = False

.Execute Replace:=wdReplaceAll

End With

DoEvents

If rHiColour <> fHiColour Then rng.Font.Emboss = False

End If

Next myGo

End If

Next hit

End If

Next FRitem

StatusBar = ""

' Restore highlight colour to normal

Options.DefaultHighlightColorIndex = oldColour

If myError < 7 Or myError > 16 Then thisRng.Select

If wasSelectedText Then rngSel.Select

totTime = Timer - timeStart

If showTime = True And totTime > 60 Then

MsgBox ((Int(10 \* totTime / 60) / 10) & \_

" minutes")

Else

If doFinalBeep = True Then

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

End If

End If

workFile.TrackRevisions = myTrack

Application.ScreenUpdating = True

Exit Sub

' Warn the user about problems that the macro has detected

myErrorReport:

If myError <= 6 Or (myError > 11 And myError < 17) \_

Or myError = 95 Then

rng.Select

Selection.Expand wdParagraph

End If

Select Case myError

Case 2: myPrompt = "No matching replace text"

Selection.MoveStart Unit:=wdParagraph, count:=-1

' Selection.Expand wdParagraph

Case 4: myPrompt = "A 'Count =' line should say 'yes' or 'no'."

Case 5: myPrompt = "Sorry, Word can't use ^p in a wildcard search." \_

& CR2 & "On Word for Mac, try [^13]." & CR2 \_

& "On Word for Windows, try ^13."

Case 6: myPrompt = "Sorry, Word can't do case insensitive " & \_

"searches with wildcards."

Case 13: myPrompt = "A 'Footnotes =' line should say 'yes' or 'no'."

Case 14: myPrompt = "An 'Endnotes =' line should say 'yes' or 'no'."

Case 15: myPrompt = "A 'Text =' line should say 'yes' or 'no'."

Case 16: myPrompt = "A 'Textboxes =' line should say 'yes' or 'no'."

Case 17: myPrompt = "You have used an unacceptable ^(something) in a search."

Case 19: myPrompt = "Unacceptable pattern match in this F&R line."

Case 21: myPrompt = "Can't find a macro called: " & rText

Case 22: myPrompt = "More Replace groups than Find groups in wildcard F&R."

Case 23: myPrompt = "^? is not a valid special character in wildcard F&R."

Case 24: myPrompt = "^ on its own is not a valid character in Replace."

Case 25: myPrompt = "The style you are trying to apply does not exist." & CR2 \_

& fText & ChrW(124) & rText

Case 95: myPrompt = "Sorry, Word can't use ^{ in a search." \_

& CR2 & "Use ^94{ instead."

Case Else: myPrompt = "Progam error; please inform Paul Bev."

End Select

thisRng.Select

MsgBox myPrompt, vbOKOnly + vbExclamation, "FRedit"

Options.DefaultHighlightColorIndex = oldColour

workFile.TrackRevisions = myTrack

theList.Activate

Application.ScreenUpdating = True

Exit Sub

' Errors that Word generates end up here

ReportIt:

Application.ScreenUpdating = True

' DoMacro call to unknown macro

If Err.Number < 0 Then myError = 21: GoTo myErrorReport

' Can't find the files it needs

' If Err.Number = 4248 Then myError = 11: GoTo myErrorReport

' Trying to run FRedit from the Zip file

' If Err.Number = 5941 Then myError = 12: GoTo myErrorReport

' Non-existent style

If Err.Number = 5834 Then myError = 25: GoTo myErrorReport

errNow = Err.Number

' If we've found the list, select it...

theList.Activate

Set rng = theList.Content

' ... and look for the current line in the list

' which is probably where the problem lies.

gottit = False

Dim myLine As Single

myLine = 1

For j = 1 To theList.Paragraphs.count

myTxt = theList.Paragraphs(j)

If Asc(myTxt) <> Asc("|") And Len(myTxt) > 2 Then

If InStr(myTxt, "|") > 0 Then

myLine = myLine + 1

Else

myLine = myLine + 0.5

End If

End If

If myLine > FRitem Then Exit For

Next j

theList.Paragraphs(j).Range.Select

If Err.Number = 5625 Or Err.Number = 5692 Then

myError = 17

GoTo myErrorReport

End If

If errNow = 5560 Then

myError = 19

GoTo myErrorReport

End If

If errNow = 5560 Or errNow = 5590 Then

myError = 19

GoTo myErrorReport

End If

' Wildcard error - too many groups

If errNow = 5623 Then myError = 22: GoTo myErrorReport

If errNow = 5692 Then myError = 23: GoTo myErrorReport

If errNow = 5624 Then myError = 24: GoTo myErrorReport

' Display Word's error message

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub

Sub UKUScount()

' Paul Beverley - Version 15.06.21

' Counts relative spellings between UK and US English

minLengthSpell = 5

countIt = True

timeStart = Timer

UKcount = 0

UScount = 0

i = ActiveDocument.Words.Count

iStart = i

StatusBar = "Spellchecking. To go: 100%"

On Error GoTo ReportIt

Application.ScreenUpdating = False

For Each wd In ActiveDocument.Words

DoEvents

If Len(wd) >= minLengthSpell And wd.Font.StrikeThrough = False Then

UKok = Application.CheckSpelling(wd, \_

MainDictionary:=Languages(wdEnglishUK).NameLocal)

USok = Application.CheckSpelling(wd, \_

MainDictionary:=Languages(wdEnglishUS).NameLocal)

If UKok <> USok Then

If UKok Then

UKcount = UKcount + 1

Else

UScount = UScount + 1

End If

wd.Select

StatusBar = "Spellchecking. To go: " & Trim(Str(Int((i / iStart) \_

\* 100))) & "% UK: " & UKcount & \_

" US: " & UScount

Debug.Print "Spellchecking. To go: " & Trim(Str(Int((i / iStart) \_

\* 100))) & "% UK: " & UKcount & \_

" US: " & UScount

End If

End If

i = i - 1

If i Mod 1000 = 0 Then StatusBar = "Spellchecking. To go: " & \_

Trim(Str(Int((i / iStart) \* 100))) & \_

"% UK: " & \_

UKcount & " US: " & UScount

Next wd

Application.ScreenUpdating = True

endTime = Timer

MsgBox "UK: " & UKcount & vbCr & vbCr & "US: " & UScount

totTime = endTime - timeStart

If countIt = True Then

If totTime > 60 Then MsgBox ((Int(10 \* totTime / 60) / 10) & \_

" minutes")

End If

Selection.HomeKey Unit:=wdStory

Beep

Exit Sub

ReportIt:

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub

Sub InstantFindDown()

' Paul Beverley - Version 12.01.21

' Finds selected text downwards

doTrim = True

addBookmark = True

butNotTheseFiles = "zzSwitchList,ComputerTools4Eds,TheMacros,5\_Library,VideoList"

If Selection.Start = Selection.End Then

Selection.Expand wdWord

For i = 1 To 3

If InStr(ChrW(8217) & " '", Right(Selection.Text, 1)) > 0 Then \_

Selection.MoveEnd , -1

Next i

End If

If Selection.Font.DoubleStrikeThrough = True Then

Selection.Expand wdParagraph

Selection.Collapse wdCollapseEnd

With Selection.Find

.ClearFormatting

.Replacement.ClearFormatting

.Wrap = wdFindStop

.Text = ""

.Font.DoubleStrikeThrough = True

.Replacement.Text = ""

.MatchWildcards = False

.MatchCase = False

.Forward = True

.Execute

End With

Exit Sub

End If

thisBit = Selection

makeWild = False

If Right(thisBit, 1) = ">" Then makeWild = True

If Left(thisBit, 1) = "<" Then makeWild = True

wordEnd = Selection.End

Selection.Collapse wdCollapseStart

myName = ActiveDocument.Name

dotPos = InStr(myName, ".")

If dotPos > 0 Then myName = Left(myName, dotPos - 1)

noMarker = (InStr(LCase(butNotTheseFiles), LCase(myName)) > 0)

If InStr(myName, "Ch") > 0 Then noMarker = True

If addBookmark = True And noMarker = False Then \_

ActiveDocument.Bookmarks.Add Name:="myTempMark2"

Selection.Start = wordEnd

If Asc(thisBit) <> 32 And doTrim Then thisBit = Trim(thisBit)

thisBit = Replace(thisBit, "^", "^^")

If makeWild = True Then

thisBit = Replace(thisBit, vbCr, "^p")

Else

thisBit = Replace(thisBit, vbCr, "^13")

End If

thisBit = Replace(thisBit, vbTab, "^t")

Selection.Start = Selection.End

hereNow = Selection.Start

With Selection.Find

.ClearFormatting

.Replacement.ClearFormatting

.Wrap = wdFindStop

.Text = thisBit

.Replacement.Text = thisBit

.MatchWildcards = makeWild

.MatchCase = False

If UCase(thisBit) = thisBit Then .MatchCase = True

.MatchWholeWord = False

.Forward = True

.Execute

End With

If Selection.Start = hereNow And \_

Selection.Find.Found = False Then Beep

' Leaves F&R dialogue in a sensible state

Selection.Find.Wrap = wdFindContinue

End Sub

Sub InstantFindUp()

' Paul Beverley - Version 13.01.21

' Finds selected text upwards

doTrim = True

addBookmark = True

butNotTheseFiles = "zzSwitchList,ComputerTools4Eds,TheMacros,5\_Library"

If Selection.Start = Selection.End Then

Selection.Expand wdWord

Do While InStr(ChrW(8217) & "' ", Right(Selection.Text, 1)) > 0

Selection.MoveEnd , -1

DoEvents

Loop

End If

DoEvents

thisBit = Selection

makeWild = False

If Right(thisBit, 1) = ">" Then makeWild = True

If Left(thisBit, 1) = "<" Then makeWild = True

wordEnd = Selection.End

Selection.Collapse wdCollapseStart

myName = ActiveDocument.Name

dotPos = InStr(myName, ".")

If dotPos > 0 Then myName = Left(myName, dotPos - 1)

noMarker = (InStr(butNotTheseFiles, myName) > 0)

If InStr(myName, "Ch") > 0 Then noMarker = True

If addBookmark = True And noMarker = False Then \_

ActiveDocument.Bookmarks.Add Name:="myTempMark2"

If Asc(thisBit) <> 32 And doTrim Then thisBit = Trim(thisBit)

thisBit = Replace(thisBit, "^", "^^")

If makeWild = True Then

thisBit = Replace(thisBit, vbCr, "^p")

Else

thisBit = Replace(thisBit, vbCr, "^13")

End If

thisBit = Replace(thisBit, vbTab, "^t")

Selection.End = Selection.Start

hereNow = Selection.Start

With Selection.Find

.ClearFormatting

.Replacement.ClearFormatting

.Wrap = wdFindStop

.Forward = False

.Text = thisBit

.Replacement.Text = thisBit

.MatchCase = False

If UCase(thisBit) = thisBit Then .MatchCase = True

.MatchWildcards = makeWild

.MatchWholeWord = False

.MatchSoundsLike = False

.Execute

End With

' Leave F&R dialogue in a sensible state

Selection.Find.Forward = True

Selection.Find.Wrap = wdFindContinue

If Selection.Start = hereNow And \_

Selection.Find.Found = False Then Beep

End Sub

Sub FindFwd()

' Paul Beverley - Version 20.12.23

' Finds next match forwards, case insensitively

mySpaces = " "

For i = 1 To 3

mySpaces = mySpaces & mySpaces

Next i

mySch = Selection.Find.Text

thisIsWild = ((InStr(mySch, "]") + InStr(mySch, "<") \_

+ InStr(mySch, ">") > 0))

If thisIsWild Then StatusBar = mySpaces & "Using WILDCARD find"

Set rng = Selection.Range.Duplicate

Selection.Collapse wdCollapseEnd

With Selection.Find

.Wrap = wdFindStop

.Forward = True

.MatchCase = False

.MatchWildcards = thisIsWild

.Execute

.Wrap = wdFindStop

End With

If Selection.Find.Found = False Then

Beep

Else

If Selection.End = 0 Then

rng.Select

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

Selection.EndKey Unit:=wdStory

With Selection.Find

.Wrap = wdFindStop

.Forward = False

.Execute

' Leave F&R dialogue in a sensible state

.Wrap = wdFindContinue

.Forward = True

End With

StatusBar = "Sorry, Word's Find facility is playing sillies!"

Else

Set rng = Selection.Range.Duplicate

Selection.Collapse wdCollapseStart

Selection.MoveUp , 1

rng.Select

End If

End If

End Sub

Sub FindBack()

' Paul Beverley - Version 08.03.23

' Next find backwards

Set rng = Selection.Range.Duplicate

Selection.Collapse wdCollapseStart

With Selection.Find

.Wrap = wdFindStop

.Forward = False

If InStr(.Text, "]") > 0 Then .MatchWildcards = True

.Execute

If .Found = False Then Beep

' Leave F&R dialogue in a sensible state

.Wrap = wdFindContinue

.Forward = True

End With

If Selection.Find.Found = False Then

Beep

Else

If Selection.Start = rng.Start Then

rng.Select

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

StatusBar = "Sorry, Word's Find facility is playing sillies!"

End If

End If

End Sub

Sub FindFwdCase()

' Paul Beverley - Version 09.01.21

' Next case-sensitive find forwards

Selection.Start = Selection.End

hereNow = Selection.End

With Selection.Find

.Wrap = wdFindStop

.Forward = True

.MatchCase = True

.Execute

End With

myStart = Selection.Start

If Selection.Start = hereNow Then

If Selection.Find.Found = False Then Beep

End If

' Now leave F&R dialogue in a sensible state

With Selection.Find

.Wrap = wdFindContinue

.Forward = True

End With

End Sub

Sub FindBackCase()

' Paul Beverley - Version 09.01.21

' Next case-sensitive find backwards

Selection.End = Selection.Start

hereNow = Selection.Start

With Selection.Find

.Wrap = wdFindStop

.Forward = False

.MatchCase = True

.Execute

End With

If Selection.Start = hereNow Then

If Selection.Find.Found = False Then Beep

End If

' Leave F&R dialogue in a sensible state

With Selection.Find

.Forward = True

.Wrap = wdFindContinue

End With

End Sub

Sub MultiSwitch()

' Paul Beverley - Version 28.01.25

' Scripted word/phrase switching

maxWords = 5

listNameIncludes = "switch"

commentNearCheck = True

' Set min number of chars for an abbreviation

minChars = 3

includeApostrophe = True

removeHighlight = True

linksOFF = True

addMarkerInList = True

notTheseChars = ",!?.)>] "

' If you want to load the SwitchList file automatically...

' then...

' On a Mac, you will need something like this:

myList = "/Users/Paul/My Documents/Macro stuff/zzSwitchList.docx"

' On Windows, you will need something like this:

myList = "C:\Documents and Settings\Paul\My Documents\zzSwitchList.docx"

CR = vbCr

CR2 = CR & CR

Dim myStr(20) As String

Dim endStr(20) As Long

On Error GoTo ReportIt

Set theDoc = ActiveDocument

' Just in case they are using "smart" cut/paste option

mySmartOpt = Options.SmartCutPaste

Options.SmartCutPaste = False

nowSearch = Selection.Find.Text

' Read the context, noting if the word is selected

wasSelected = (Selection.End > Selection.start)

If wasSelected = True Then maxWords = 1

myTest = Selection.Text

If Asc(myTest) = 13 Then Selection.MoveLeft , 1

If InStr(notTheseChars, myTest) > 0 Then Selection.MoveLeft , 1

Selection.Expand wdWord

Do While InStr(ChrW(8217) & "' ", Right(Selection.Text, 1)) > 0

Selection.MoveEnd , -1

DoEvents

Loop

okChars = ".,;:" & Chr(11) & vbCr & ChrW(8211) & ChrW(8212) & ChrW(8221)

If InStr(okChars, Left(Selection, 1)) > 0 Then

Selection.Collapse wdCollapseStart

Selection.MoveLeft , 1

Selection.Expand wdWord

End If

isAbbrev = Not (Len(Selection) > minChars)

startNow = Selection.start

Set startRng = Selection.Range.Duplicate

iMax = maxWords

Set rng = Selection.Range.Duplicate

rng.Collapse wdCollapseStart

rng.MoveEnd Unit:=wdWord, count:=maxWords

numTCs = rng.Revisions.count

If numTCs > 0 Then

rng.Revisions.AcceptAll

rng.MoveEnd Unit:=wdWord, count:=maxWords

End If

wdsLeft = rng.Words.count

If maxWords > wdsLeft Then maxWords = wdsLeft

If Asc(rng.Words(wdsLeft)) = 13 Then maxWords = maxWords - 1

For i = 1 To maxWords

endStr(i) = rng.Words(i).End

If rng.Words(i) = vbCr Then

maxWords = i - 1

Exit For

End If

If Right(rng.Words(i), 1) = " " Then endStr(i) = endStr(i) - 1

myStr(i) = Trim(Left(rng, endStr(i) - rng.start))

DoEvents

Next i

If numTCs > 0 Then WordBasic.EditUndo

Selection.Collapse wdCollapseStart

iMax = maxWords

gottaList = False

For Each myDoc In Application.Documents

thisName = myDoc.Name

If InStr(LCase(thisName), LCase(listNameIncludes)) > 0 Then

Set theList = Documents(thisName)

gottaList = True

Exit For

End If

Next myDoc

' Find the zzSwitchList file

If gottaList = False Then

Documents.Open fileName:=myList

Set theList = ActiveDocument

theDoc.Activate

End If

Set rng = theList.Content

' Check if list has LFs

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Wrap = wdFindContinue

.Text = ChrW(11)

.Replacement.Text = ""

.Execute

DoEvents

If .Found = True Then

Beep

rng.start = rng.start - 1

rng.Collapse wdCollapseStart

rng.Expand wdWord

rng.HighlightColorIndex = wdYellow

rng.Collapse wdCollapseEnd

rng.Select

MsgBox "The switch list must use paragraphs, not line breaks!"

ActiveWindow.ActivePane.View.ShowAll = True

rng.Find.Text = nowSearch

Exit Sub

End If

End With

' Just in case the switch list has any URLs

If linksOFF = True Then

rng.Fields.Unlink

End If

lastTwo = Right(rng, 2)

If lastTwo <> CR2 Then rng.InsertAfter Text:=CR2

firstTwo = Left(rng, 2)

If firstTwo <> CR2 Then rng.InsertBefore Text:=CR2

rng.start = 0

rng.End = theList.Content.End

' Try to locate the selected text in the switch list

allWords = rng.Text

lfPos = InStr(allWords, ChrW(11))

If lfPos > 0 Then

rng.start = lfPos - 2

rng.End = lfPos + 1

rng.Select

Beep

Application.Run "ShowFormatting"

MsgBox "The switch list must use paragraphs, not line breaks!"

Options.SmartCutPaste = mySmartOpt

Selection.Find.Text = nowSearch

Exit Sub

End If

For i = iMax To 1 Step -1

myPos = InStr(allWords, CR2 & Replace(myStr(i), CR, "") & CR)

If myPos > 0 Then

allWords = Mid(allWords, myPos + 1)

gottaMatch = True

numWords = i

'Display the options

numAlts = 1

endPos = InStr(allWords, CR2)

allWords = Left(allWords, endPos + 2)

altText = Split(allWords, Chr(13))

j = 0

Do

j = j + 1

Loop Until Len(altText(j)) = 0 Or j = 20

numAlts = j - 2

' We now have j = number of options

' altText contains all numAlts texts

Exit For

Else

gottaMatch = False

End If

Next i

' If it's not found at all, give up

If gottaMatch = False Then

myWd = myStr(1)

Do

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^p^p" & myWd

.Wrap = wdFindContinue

.Replacement.Text = ""

.Forward = True

.MatchCase = True

.MatchWildcards = False

.Execute

.MatchCase = False

End With

If Len(myWd) > 1 Then myWd = Left(myWd, Len(myWd) - 1)

Loop Until rng.Find.Found = True Or Len(myWd) = 1

rng.Collapse wdCollapseStart

rng.Select

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

Options.SmartCutPaste = mySmartOpt

Selection.Find.Text = nowSearch

Debug.Print "|" & myStr(1) & "|"

myPrompt = "Can't find """ & myStr(1) & """"

myResponse = MsgBox(myPrompt, vbOKOnly, "MultiSwitch")

theDoc.Activate

Exit Sub

End If

'Offer j optional texts to the user

If numAlts > 1 And wasSelected = False Then

For i = 1 To numAlts

myPrompt = myPrompt & i & ": " & altText(i + 1) & CR

Next

Beep

myResponse = InputBox(myPrompt, "MultiSwitch")

myChoice = Val(myResponse)

If myChoice > numAlts Or myChoice = 0 Then

Options.SmartCutPaste = mySmartOpt

Selection.Find.Text = nowSearch

Exit Sub

End If

Else

myChoice = 1

End If

rng.start = myPos + 2

rng.Collapse wdCollapseStart

rng.Expand wdParagraph

For i = 1 To myChoice

rng.Collapse wdCollapseEnd

rng.Expand wdParagraph

Next i

' Copy chosen line (check first for format change symbol)

useFormat = False

rng.MoveEnd , -1

myNewItem = rng

Dim gotFormat As Boolean

gotFormat = rng.Font.Bold Or rng.Font.Italic Or rng.Font.SmallCaps \_

Or rng.Font.Underline Or rng.Font.Superscript Or rng.Font.Subscript \_

Or rng.InlineShapes.count > 0

If rng.Text = "" Then Beep: Exit Sub

If AscW(rng) = 172 Or gotFormat Then

useFormat = True

If AscW(rng) = 172 Then rng.MoveStart , 1

rng.Copy

End If

noTrack = (rng.Font.StrikeThrough)

startRng.Select

Selection.End = endStr(numWords)

' Check if too near to a comment

If Selection.Information(wdInCommentPane) = 0 And \_

Selection.Information(wdInFootnote) = 0 And \_

Selection.Information(wdInEndnote) = 0 Then

numCmts = ActiveDocument.Comments.count

If numCmts > 0 And commentNearCheck = True Then

' Find scope of (range covered by) the next comment

Set rng = Selection.Range.Duplicate

rng.Collapse wdCollapseStart

rng.start = 0

cmtNumber = rng.Comments.count + 1

If cmtNumber <= numCmts Then

Set nextCmtRange = ActiveDocument.Comments(cmtNumber).Scope

' Does the scope of that comment overlap the current word?

Set rng = Selection.Range.Duplicate

rng.Expand wdWord

If rng.End > nextCmtRange.start Then

Beep

rng.Select

myResponse = MsgBox("Selection contains a comment. Word might crash!" \_

& CR2 & "Please use a manual change.", \_

vbOKOnly, "MultiSwitch")

Options.SmartCutPaste = mySmartOpt

Selection.Find.Text = nowSearch

Exit Sub

End If

End If

End If

End If

With ActiveWindow.View.RevisionsFilter

myMarkup = .Markup

myView = .View

End With

myTrack = ActiveDocument.TrackRevisions

If noTrack Then ActiveDocument.TrackRevisions = False

If useFormat = True Then

Selection.Delete

Selection.Paste

Else

If removeHighlight = True Then

myTrack = ActiveDocument.TrackRevisions

ActiveDocument.TrackRevisions = False

Selection.Range.HighlightColorIndex = wdNoHighlight

ActiveDocument.TrackRevisions = myTrack

End If

Selection.Text = myNewItem

End If

Selection.start = startNow

textEnd = Selection.End

If InStr(Selection, "^p") > 0 Then

numCRs = 0

oldFind = Selection.Find.Text

With Selection.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^^p"

.Wrap = wdFindContinue

.Replacement.Text = ""

.Forward = True

.MatchCase = False

.MatchWildcards = False

.Execute

End With

Do While Selection.Find.Found = True And Selection.start < textEnd

Selection.TypeText Text:=vbCr

Selection.Find.Execute

numCRs = numCRs + 1

Loop

Selection.start = startNow

Selection.End = textEnd - numCRs

Selection.Find.Text = oldFind

End If

If InStr(Selection, "^t") > 0 Then

numTabs = 0

oldFind = Selection.Find.Text

With Selection.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^^t"

.Wrap = wdFindContinue

.Replacement.Text = ""

.Forward = True

.MatchCase = False

.MatchWildcards = False

.Execute

End With

Do While Selection.Find.Found = True And Selection.start < textEnd

Selection.TypeText Text:=vbTab

Selection.Find.Execute

numTabs = numTabs + 1

Loop

Selection.start = startNow

Selection.End = textEnd - numCRs - numTabs

Selection.Find.Text = oldFind

End If

If Asc(Selection) = Asc("!") Then

myLen = Len(Selection)

Selection.End = Selection.start + 1

Selection.start = Selection.start - 1

Selection.Delete

Selection.MoveEnd , myLen - 1

End If

tildePos = InStr(Selection, "~")

Selection.Font.StrikeThrough = False

If tildePos = 0 Then

If isAbbrev = True Then

Selection.Collapse wdCollapseEnd

Else

Selection.Collapse wdCollapseStart

End If

Else

Do While InStr(Selection, "~") > 0

Selection.MoveEnd , -1

DoEvents

Loop

Selection.Collapse wdCollapseEnd

Selection.MoveEnd 1

Selection.Delete

End If

If addOriginal = True Then

Selection.Expand wdWord

Do While InStr(ChrW(8217) & "' ", Right(Selection.Text, 1)) > 0

Selection.MoveEnd , -1

DoEvents

Loop

Selection.Collapse wdCollapseEnd

Selection.TypeText Text:=" (" & myStr(1) & ")"

End If

If noTrack Then

ActiveDocument.TrackRevisions = myTrack

With ActiveWindow.View.RevisionsFilter

.Markup = myMarkup

.View = myView

End With

End If

Options.SmartCutPaste = mySmartOpt

Selection.Find.Text = "^p^p" & myStr(1)

Exit Sub

ReportIt:

myErr = Err.Number

If myErr = 4160 Then

DoEvents

gottaList = False

Resume Next

Else

If myErr = 438 Then Beep: Exit Sub

If myErr = 5174 Then

Beep

myPrompt = "Please open your switch list file: " & myListName \_

& CR2 & CR & "I did try looking for file: " & CR2 & myList

myResponse = MsgBox(myPrompt, vbQuestion + vbOKOnly, "MultiSwitch")

Exit Sub

End If

DoEvents

Resume

End If

End Sub

Sub MacroLauncher()

' Paul Beverley - Version 12.05.22

' Offers a list of macros to launch (was called "MacroMenu")

myList1 = "a=AccentAlyse, d=DocAlyse, h=HyphenAlyse"

myList2 = "w=WordPairAlyse, p=ProperNounAlyse, i=IZISCount"

myList3 = "z=IStoIZ, s=IZtoIS, l=SpellingErrorLister"

myList4 = "c=CopyTextSimple, m=MultiFileText, f=FRedit"

myList5 = "e=SpellingErrorHighlighter, u=UKUSCount"

' myList5 = "e=SpellingErrorHighlighter, u=UKUSCount, g=MegAlyse"

myList = "," & myList1 & "," & myList2 & "," & myList3 & "," \_

& myList4 & "," & myList5

myList = Replace(myList, ",,", ",")

mcr = Split(myList, ",")

myPrompt = ""

myCodes = ""

numItems = UBound(mcr)

For i = 1 To numItems

mcr(i) = Trim(mcr(i))

myCode = Left(mcr(i), 1)

myCodes = myCodes & myCode

mName = Mid(mcr(i), 3)

myPrompt = myPrompt & Trim(Str(i)) & " " & ChrW(8211) & " "

myPrompt = myPrompt & mName & " (" & myCode & ")" & vbCr

Next i

Do

myResponse = UCase(InputBox(myPrompt, "MacroLauncher"))

myNum = Val(myResponse)

If myResponse = "" Then Exit Sub

If myNum = 0 Then

myNum = InStr(UCase(myCodes), Left(myResponse, 1))

End If

Loop Until myNum > 0 And myNum < numItems + 1

mName = Mid(mcr(myNum), 3)

Application.Run MacroName:=mName

End Sub

Sub WordPairAlyse()

' Paul Beverley - Version 05.01.21

' Creates a file of all the adjacent word pairs

' Ignore these words

nonWords = "a,as"

addAtestPair = True

Set FUT = ActiveDocument

aT = LCase(FUT.Content.Text)

If addAtestPair = True Then

aT = "aatest word and aatestword " & aT

End If

doingSeveralMacros = (InStr(FUT.Name, "zzTestFile") > 0)

If doingSeveralMacros = False Then

myResponse = MsgBox(" WordPairAlyse" & vbCr & vbCr & \_

"Find word pairs?", vbQuestion \_

+ vbYesNoCancel, "WordPairAlyse")

If myResponse <> vbYes Then Exit Sub

End If

startTime = Timer

chs = " , . ! : ; [ ] { } ( ) / \ + "

chs = chs & ChrW(8220) & " "

chs = chs & ChrW(8221) & " "

chs = chs & ChrW(8201) & " "

chs = chs & ChrW(8222) & " "

chs = chs & ChrW(8217) & " "

chs = chs & ChrW(8216) & " "

chs = chs & ChrW(8212) & " "

chs = chs & ChrW(8722) & " "

chs = chs & vbCr & " "

chs = chs & vbTab & " "

chs = " " & chs & " "

chs = Replace(chs, " ", " ")

chs = Left(chs, Len(chs) - 1)

chars = Split(chs, " ")

For i = 1 To UBound(chars)

aT = Replace(aT, chars(i), " " & chars(i) & " ")

Next i

' Remove all non-words

nonWords = "," & nonWords & ","

nonWords = Replace(nonWords, ",,", ",")

nonWords = Left(nonWords, Len(nonWords) - 1)

wd = Split(nonWords, ",")

Set rng = ActiveDocument.Content

For i = 1 To UBound(wd)

aT = Replace(aT, " " & wd(i) & " ", " ")

DoEvents

Next i

aT = Replace(aT, " ", " ")

Documents.Add

Selection.Text = " " & aT

Set rng = ActiveDocument.Content

Selection.HomeKey Unit:=wdStory

Set rng = ActiveDocument.Content

aT = LCase(rng.Text)

myTot = Len(aT)

If Asc(aT) = 32 Then

ptr = 2

Else

ptr = 1

End If

ptrWas = ptr

Do

ch = Mid(aT, ptr, 1)

' Debug.Print ch & "|"

ptr = ptr + 1

Loop Until ch = " "

w2 = Mid(aT, ptrWas, ptr - ptrWas - 1)

' Debug.Print w2 & "|"

allChkd = " "

myResult = ""

Do

w1 = w2

ptrWas = ptr

Do

ch = Mid(aT, ptr, 1)

ptr = ptr + 1

Loop Until ch = " "

w2 = Mid(aT, ptrWas, ptr - ptrWas - 1)

If UCase(w1) <> w1 And UCase(w2) <> w2 Then

oneWd = w1 & w2

chk = " " & oneWd & " "

If InStr(allChkd, chk) = 0 Then

' Check it!

If InStr(aT, chk) > 0 Then

numSingle = Len(Replace(aT, chk, chk & "!")) - myTot

chk2 = " " & w1 & " " & w2 & " "

numPair = Len(Replace(aT, chk2, chk2 & "!")) - myTot

myResult = myResult & w1 & " " & w2 & " . . " & \_

Trim(Str(numPair)) & vbCr

myResult = myResult & oneWd & " . . " & \_

Trim(Str(numSingle)) & vbCr & vbCr

Debug.Print Trim(Str(Int((myTot - ptr) / 6000))) \_

& ",000 to go" & " " & w1 & " " & w2

StatusBar = Trim(Str(Int((myTot - ptr) / 6000))) \_

& ",000 to go" & " " & w1 & " " & w2

End If

allChkd = allChkd & oneWd & " "

End If

End If

DoEvents

Loop Until InStr(Mid(aT, ptr), " ") = 0

Selection.WholeStory

Selection.Delete

If Len(myResult) > 0 Then

Selection.Text = myResult

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^p^p"

.Replacement.Text = "zczc"

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

.Text = "^p"

.Replacement.Text = ":"

.Execute Replace:=wdReplaceAll

.Text = "zczc"

.Replacement.Text = "^p"

.Execute Replace:=wdReplaceAll

End With

Selection.WholeStory

Selection.Sort SortOrder:=wdSortOrderAscending

With rng.Find

.Text = "^p"

.Replacement.Text = "^p^p"

.Execute Replace:=wdReplaceAll

.Text = ":"

.Replacement.Text = "^p"

.Execute Replace:=wdReplaceAll

End With

Selection.Start = 0

Selection.End = 3

Selection.Delete

Else

Selection.TypeText "No word pairs found" & vbCr

End If

Selection.HomeKey Unit:=wdStory

Selection.TypeText "Word pair (possible) inconsistencies" & vbCr

ActiveDocument.Paragraphs(1).Style = \_

ActiveDocument.Styles(wdStyleHeading1)

timNow = Timer

timGone = timNow - startTime

m = Int(timGone / 60)

s = Int(timGone) - m \* 60

If doingSeveralMacros = False Then

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.3

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.3

Beep

MsgBox "Total time:" & Str(m) & " m " & Str(s) & " s"

Else

FUT.Activate

End If

End Sub

Sub CopyTextSimple()

' Paul Beverley - Version 03.02.20

' Creates a text-only copy, with some features preserved

' Used to be called CopyTextWithSomeFeatures

doItalic = True

doBold = True

doSub = True

doSuper = True

myColour = wdGray25

myComments = ""

Set rngOld = ActiveDocument.Content

Documents.Add

Set rng = ActiveDocument.Content

rng.FormattedText = rngOld.FormattedText

rng.Revisions.AcceptAll

If numberCmnts > 0 Then ActiveDocument.DeleteAllComments

wds1 = rng.Words.Count

rng.Font.Hidden = False

wds2 = rng.Words.Count

If wds2 <> wds1 Then

WordBasic.EditUndo

myResponse = MsgBox("Original contains " & wds2 - wds1 & \_

" words of hidden text. Include it?", \_

vbQuestion + vbYesNoCancel, "CopyText")

If myResponse = vbCancel Then Exit Sub

If myResponse = vbYes Then

ActiveDocument.ActiveWindow.View.ShowHiddenText = True

oldColour = Options.DefaultHighlightColorIndex

Options.DefaultHighlightColorIndex = myColour

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Hidden = True

.Wrap = wdFindContinue

.Replacement.Font.Hidden = False

.Replacement.Highlight = True

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

End With

ActiveDocument.ActiveWindow.View.ShowHiddenText = False

End If

End If

numNotes = ActiveDocument.Endnotes.Count

If numNotes > 0 Then

myComments = myComments & "| endnotes = yes (" \_

& Trim(Str(numNotes)) & ")" & vbCr

Set rng = ActiveDocument.Content

rng.Collapse wdCollapseEnd

rng.InsertAfter Text:=vbCr & "Endnotes:" & vbCr & vbCr

rng.Collapse wdCollapseEnd

rng.FormattedText = \_

ActiveDocument.StoryRanges(wdEndnotesStory).FormattedText

' Delete all notes

For j = numNotes To 1 Step -1

ActiveDocument.Endnotes(j).Delete

Next j

End If

numNotes = ActiveDocument.Footnotes.Count

If numNotes > 0 Then

myComments = myComments & "| footnotes = yes (" \_

& Trim(Str(numNotes)) & ")" & vbCr

Set rng = ActiveDocument.Content

rng.Collapse wdCollapseEnd

rng.InsertAfter Text:=vbCr & "Footnotes:" & vbCr & vbCr

rng.Collapse wdCollapseEnd

rng.FormattedText = \_

ActiveDocument.StoryRanges(wdFootnotesStory).FormattedText

' Delete all notes

For j = numNotes To 1 Step -1

ActiveDocument.Footnotes(j).Delete

Next j

End If

' copy all the textboxes to the end of the text

shCount = ActiveDocument.Shapes.Count

tbCount = 0

If shCount > 0 Then

Selection.EndKey Unit:=wdStory

Selection.TypeText vbCr & "Textboxes:" & vbCr & vbCr

For j = 1 To shCount

Set shp = ActiveDocument.Shapes(j)

If shp.Type <> 24 And shp.Type <> 3 Then

If shp.TextFrame.hasText Then

Set rng = shp.TextFrame.TextRange

If Len(rng.Text) > 1 Then

tbCount = tbCount + 1

Selection.FormattedText = rng.FormattedText

Selection.EndKey Unit:=wdStory

End If

End If

End If

Next

If tbCount > 0 Then myComments = myComments & \_

"| textboxes = yes (" & Trim(Str(tbCount)) \_

& ")" & vbCr

End If

' Add a newline for safety

Selection.TypeText vbCr

Selection.HomeKey Unit:=wdStory

If myComments > "" Then Selection.TypeText myComments & vbCr

' Watch out for hard spaces and spaced dots for ellipses

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^s"

.Wrap = wdFindContinue

.Replacement.Text = " "

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ". . ."

.Wrap = wdFindContinue

.Replacement.Text = "…"

.Execute Replace:=wdReplaceAll

End With

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Highlight = True

.Wrap = wdFindContinue

.Replacement.Text = "hkhk^&khkh"

.MatchCase = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

If doItalic = True Then

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Italic = True

.Wrap = wdFindContinue

.Replacement.Text = "zczc^&czcz"

.MatchCase = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

End If

If doBold = True Then

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Bold = True

.Wrap = wdFindContinue

.Replacement.Text = "jqjq^&qjqj"

.MatchCase = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

End If

If doSub = True Then

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Subscript = True

.Wrap = wdFindContinue

.Replacement.Text = "xbxb^&bxbx"

.MatchCase = True

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

End If

If doSuper = True Then

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Superscript = True

.Wrap = wdFindContinue

.MatchCase = True

.Replacement.Text = "xsxs^&sxsx"

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

End If

' Copy pure text after the end of the formatted text

EndNow = ActiveDocument.range.End

Set rngNew = ActiveDocument.Content

rngNew.Collapse wdCollapseEnd

rngNew.Text = rng.Text

' Delete the formatted text

Set rng = ActiveDocument.Content

rng.Collapse wdCollapseStart

' One character less, to avoid the safety newline

rng.End = EndNow - 1

rng.Delete

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[^12^14]"

.Wrap = wdFindContinue

.Replacement.Text = "^p"

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

If doSuper = True Then

Set rngNew = ActiveDocument.Content

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "xsxs(\*)sxsx"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Superscript = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "XSXS(\*)SXSX"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Superscript = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

End If

If doSub = True Then

Set rngNew = ActiveDocument.Content

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "xbxb(\*)bxbx"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Subscript = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "XBXB(\*)BXBX"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Subscript = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

End If

If doBold = True Then

Set rngNew = ActiveDocument.Content

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "jqjq(\*)qjqj"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Bold = True

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "JQJQ(\*)QJQJ"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Bold = True

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

End If

If doItalic = True Then

Set rngNew = ActiveDocument.Content

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "zczc(\*)czcz"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Italic = True

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "ZCZC(\*)CZCZ"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Font.Italic = True

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

End If

' Restore Highlight

Set rngNew = ActiveDocument.Content

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "hkhk(\*)khkh"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Highlight = True

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "HKHK(\*)KHKH"

.Wrap = wdFindContinue

.Replacement.Text = "\1"

.Replacement.Highlight = True

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

Options.DefaultHighlightColorIndex = oldColour

' Tidy up picture markers

Set rngNew = ActiveDocument.Content

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "^1"

.Wrap = wdFindContinue

.Replacement.Text = ""

.MatchWildcards = False

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[^13]{3,}"

.Wrap = wdFindContinue

.Replacement.Text = "^p^p"

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[hkjzcqxbsHKJZCQXBS]{4,}"

.Wrap = wdFindContinue

.Replacement.Text = ""

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

If tbCount = 0 Then

With rngNew.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "Textboxes:"

.Wrap = wdFindContinue

.Replacement.Text = ""

.MatchCase = True

.MatchWildcards = True

.Execute Replace:=wdReplaceAll

End With

End If

Beep

End Sub

Sub MultiFileText()

' Paul Beverley - Version 08.11.20

' Collects text plus simple formatting from multiple files

spaceEquations = True

listOff = True

acceptTCs = True

addFilename = True

convertMathsItems = True

remindAboutCancel = False

On Error GoTo ReportIt

CR2 = vbCr & vbCr

Dim allMyFiles(200) As String

Set rng = ActiveDocument.Content

myExtent = 250

' First find if this is a Mac or a PC!

myFullName = ActiveDocument.FullName

myName = ActiveDocument.Name

myFolder = Replace(myFullName, myName, "")

myDelimiter = Application.PathSeparator

If myDelimiter = "/" Then

InAMac = True

Else

InAMac = False

End If

If rng.End - rng.Start > myExtent Then rng.End = rng.Start + myExtent

If InStr(LCase(rng.Text), ".doc") = 0 And InStr(LCase(rng.Text), ".rtf") = 0 Then

' If not a file list then navigate to the required folder

If remindAboutCancel = True Then myResponse = \_

MsgBox("Navigate to the required folder; then press 'Cancel'" \_

, , "Multifile Text Collection")

docCount = Documents.Count

Dialogs(wdDialogFileOpen).Show

If Documents.Count > docCount Then ActiveDocument.Close

dirPath = CurDir()

ChDir dirPath

' Read the names of all the files in this directory

myFile = Dir(CurDir() & myDelimiter)

Documents.Add

numFiles = 0

Do While myFile <> ""

If InStr(LCase(myFile), ".doc") > 0 Or InStr(LCase(myFile), ".rtf") > 0 Then

Selection.TypeText myFile & vbCr

numFiles = numFiles + 1

End If

myFile = Dir()

Loop

' Now sort the file list (only actually needed for Macs)

Selection.WholeStory

Selection.Sort SortOrder:=wdSortOrderAscending, \_

SortFieldType:=wdSortFieldAlphanumeric

Selection.EndKey Unit:=wdStory

Selection.TypeParagraph

Selection.HomeKey Unit:=wdStory

Selection.TypeText dirPath

' Go back until you hit myDelimiter

Selection.MoveStartUntil cset:=myDelimiter, Count:=wdBackward

dirName = Selection

Selection.HomeKey Unit:=wdStory

myResponse = MsgBox("Collect unformatted text from ALL the files in" & \_

" directory: " & dirName & "?", vbQuestion + vbYesNoCancel, \_

"Multifile Text Collection")

If myResponse <> vbYes Then Exit Sub

Else

myResponse = MsgBox("Collect unformatted text from the files listed here?", \_

vbQuestion + vbYesNoCancel, "Multifile Text Collection")

If myResponse <> vbYes Then Exit Sub

End If

If InAMac = False Then

ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal

Application.Move Left:=20, Top:=300

Application.Resize Width:=1000, Height:=200

End If

' Pick up the folder name and the filenames from the file list

numFiles = 0

myFolder = ""

For Each myPara In ActiveDocument.Paragraphs

myPara.Range.Select

Selection.MoveEnd , -1

lineText = Selection

If myFolder = "" Then

myFolder = lineText

Selection.Collapse wdCollapseEnd

Selection.MoveStartUntil cset:=myDelimiter, Count:=wdBackward

Selection.MoveStart , -1

myDelimiter = Left(Selection, 1)

Else

thisFile = lineText

If Len(thisFile) > 2 Then

If Left(thisFile, 1) <> "|" Then

numFiles = numFiles + 1

allMyFiles(numFiles) = thisFile

End If

End If

End If

Next myPara

Selection.HomeKey Unit:=wdStory

Set alltextDoc = Documents.Add

Set alltextRng = ActiveDocument.Content

Selection.TypeText "Loading first file" & vbCr & vbCr

Selection.HomeKey Unit:=wdStory

ActiveDocument.Paragraphs(1).Range.Bold = True

ActiveDocument.Paragraphs(1).Range.Font.Size = 40

If InAMac = False Then

ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal

Application.Move Left:=20, Top:=2

Application.Resize Width:=1000, Height:=250

End If

gotLanguage = False

For i = 1 To numFiles

alltextDoc.Paragraphs(1).Range = allMyFiles(i) & vbCr

' Get the folder name, and then the text for the files in the list

thisFile = myFolder & myDelimiter & allMyFiles(i)

Set myDoc = Application.Documents.Open(FileName:=thisFile)

If InAMac = False Then

ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal

Application.Move Left:=20, Top:=300

Application.Resize Width:=1000, Height:=200

End If

If gotLanguage = False Then

Selection.MoveEnd , 1

myLanguage = Selection.Range.LanguageID

gotLanguage = True

End If

StatusBar = allMyFiles(i)

numEqns = ActiveDocument.OMaths.Count

If numEqns > 0 And spaceEquations = True Then

For Each myMath In ActiveDocument.OMaths

myMath.Range.InsertBefore Text:=" "

DoEvents

numEqns = numEqns - 1

Debug.Print numEqns

Next myMath

End If

DoEvents

If acceptTCs = True Then myDoc.Revisions.AcceptAll

myDoc.TrackRevisions = False

If listOff = True Then myDoc.ConvertNumbersToText

If ActiveDocument.Endnotes.Count > 0 Then

Set thisDocRange = myDoc.Content

thisDocRange.Collapse wdCollapseEnd

thisDocRange.FormattedText = \_

myDoc.StoryRanges(wdEndnotesStory).FormattedText

End If

If ActiveDocument.Footnotes.Count > 0 Then

Set thisDocRange = myDoc.Content

thisDocRange.Collapse wdCollapseEnd

thisDocRange.FormattedText = \_

myDoc.StoryRanges(wdFootnotesStory).FormattedText

End If

' copy all the textboxes to the end of the text

shCount = myDoc.Shapes.Count

If shCount > 0 Then

Selection.EndKey Unit:=wdStory

Selection.TypeText vbCr & vbCr

For j = 1 To shCount

Set shp = ActiveDocument.Shapes(j)

If shp.Type <> 24 And shp.Type <> 3 Then

If shp.TextFrame.HasText Then

Set rng = shp.TextFrame.TextRange

Selection.FormattedText = rng.FormattedText

Selection.EndKey Unit:=wdStory

End If

End If

Next

End If

Selection.HomeKey Unit:=wdStory

If addFilename = True Then Selection.TypeText CR2 & "[[[[ " & \_

allMyFiles(i) & " ]]]]]" & CR2

If convertMathsItems = True Then

Application.Visible = False

numMaths = ActiveDocument.OMaths.Count

If numMaths > 0 Then

For Each myMath In ActiveDocument.OMaths

myMath.Range.Select

eqText = Replace(Selection.Text, vbCr, " ")

Selection.MoveStart , -1

Selection.Delete

startHere = Selection.Start

Selection.TypeText eqText

Selection.Start = startHere

Next myMath

End If

Application.Visible = True

End If

myDoc.Fields.Unlink

' Accept all track changes

myDoc.Range.Revisions.AcceptAll

myDoc.TrackRevisions = False

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Italic = True

.Wrap = wdFindContinue

.MatchWildcards = False

.MatchCase = True

.Replacement.Text = "zccz^&pqqp"

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Bold = True

.Replacement.Text = "bqqb^&zwvf"

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Superscript = True

.Replacement.Text = "yxzu^&qiwv"

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = ""

.Font.Subscript = True

.Replacement.Text = "xhwc^&yvxz"

.Execute Replace:=wdReplaceAll

End With

' Hide the screen to avoid mouse movement

Application.Visible = False

Set rng = ActiveDocument.Content

alltextRng.End = alltextDoc.Range.End

alltextRng.Start = alltextDoc.Range.End

alltextRng.Text = rng.Text

myDoc.Close SaveChanges:=wdDoNotSaveChanges

' Show the screen again

Application.Visible = True

Next i

alltextDoc.Paragraphs(1).Range.Select

Selection.Delete

alltextDoc.Paragraphs(2).Range.Delete

Set rng = alltextDoc.Content

rng.LanguageID = myLanguage

rng.NoProofing = False

Set rng = ActiveDocument.Content

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "yxzu(\*)qiwv"

.MatchWildcards = True

.MatchCase = False

.Replacement.Text = "^&"

.Replacement.Font.Superscript = True

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "xhwc(\*)yvxz"

.Replacement.Text = "^&"

.Replacement.Font.Subscript = True

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "bqqb(\*)zwvf"

.Replacement.Text = "^&"

.Replacement.Font.Bold = True

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "zccz(\*)pqqp"

.Replacement.Text = "^&"

.Replacement.Font.Italic = True

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "zccz"

.MatchWildcards = False

.MatchCase = False

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "yxzu"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "bqqb"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "zwvf"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "qiwv"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "pqqp"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "xhwc"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

.ClearFormatting

.Replacement.ClearFormatting

.Text = "yvxz"

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

' Remove optional hyphens

.ClearFormatting

.Replacement.ClearFormatting

.Text = Chr(172)

.MatchWildcards = False

.MatchCase = True

.Replacement.Text = ""

.Execute Replace:=wdReplaceAll

End With

Set rng = ActiveDocument.Content

totChars = rng.End

' Add one '[' for each file, to check they are all there

With rng.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[[[[ "

.Wrap = wdFindContinue

.MatchWildcards = False

.MatchWholeWord = False

.Replacement.Text = "[[[[[ "

.Execute Replace:=wdReplaceAll

End With

ActiveDocument.Paragraphs(3).Range.Bold = True

Set rng = ActiveDocument.Content

totCharsNow = rng.End

If InAPC = True Then

ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal

Application.Resize Width:=1000, Height:=500

End If

Beep

MsgBox "Please now WAIT until the flashing cursor returns." & vbCr \_

& vbCr & "Not doing so can cause Word to crash." & vbCr & \_

vbCr & " If unsure, only try pressing the cursor right key."

With Selection.Find

.ClearFormatting

.Replacement.ClearFormatting

.Text = "[[[[["

.Wrap = wdFindContinue

.MatchWildcards = False

.MatchWholeWord = False

.Replacement.Text = ""

.Execute

End With

If totCharsNow - totChars <> numFiles Then

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

MsgBox "Warning: All files might not have been included."

End If

Exit Sub

ReportIt:

Application.Visible = True

On Error GoTo 0

Resume

End Sub

Sub QuickSort(arr As Variant, ByVal low As Long, ByVal high As Long)

' Paul Beverley - Version 21.12.24

' Sorts an array of text at high speed

Dim i As Long

Dim j As Long

Dim pivot As String

Dim temp As String

i = low

j = high

pivot = arr((low + high) \ 2)

Do While i <= j

Do While arr(i) < pivot

i = i + 1

Loop

Do While arr(j) > pivot

j = j - 1

Loop

If i <= j Then

temp = arr(i)

arr(i) = arr(j)

arr(j) = temp

i = i + 1

j = j - 1

End If

Loop

If low < j Then QuickSort arr, low, j

If i < high Then QuickSort arr, i, high

End Sub