```
Public GETOPTION_RET_VAL As Object 'Passed back to the function from the UserForm

Private Function Item_Duplicate_Check(theFindItem As String, theHTSCounter As Long) As Integer
    Dim htsCounter As Long
    htsCounter = theHTSCounter + 1 '' so we dont go back through and search previously searched rows (+1 is ↙
     so we dont re-register the start of recursion again)

    Dim duplicateCounter As Integer '' counter to register duplicates
    duplicateCounter = 0

    Dim findItem As String '' item we are actually looking for
    findItem = theFindItem

    ''searching through HTS Code List to find the current item number from Invoice (while rows not blank,  ↙
    and while recursive loop has not been started)
    ''once the recursion has started, we eliminate the need to look further for the nth occurance of      ↙
    findItem
    While ((Sheets("HTS Code List").Cells(htsCounter, 3) = "" And Sheets("HTS Code List").Cells(htsCounter, ↙
     4) = "" And Sheets("HTS Code List").Cells(htsCounter, 12) = "") And (duplicateCounter = 0))
        ''recursively checking through to see if there are duplicates of item number
        If (Sheets("HTS Code List").Cells(htsCounter, 3) = findItem) Then
            ''at this point, there are duplicates present.  we just need to find all of them and kick back ↙
    notice to upper loops
            duplicateCounter = 1 + Item_Duplicate_Check(findItem, htsCounter)


            'Dim Ops(1 To 12) As String '' going to need to have all of the suppliers listed here
            ''   Create an array of month names
            'For i = 1 To 12
            '    Ops(i) = Format(DateSerial(1, i, 1), "mmmm")
            'Next i
            'UserChoice = GetOption(Ops, 1, "Select a month")
            'If UserChoice <> False Then MsgBox UserChoice
            'GETOPTION_RET_VAL should be equal to the name of the supplier that should be used



        Else
            duplicateCounter = 0
        End If
        htsCounter = htsCounter + 1
    End While


    Item_Duplicate_Check = duplicateCounter
End Function

Private Sub TotalButton_Click()

    Call ClearTotals() ''clear worksheet
    Call Delete_Blank_HTS_Rows() ''delete all blank rows from hts list

    ''arrays to hold final values to be displayed on worksheet
    Dim theHTSCode(20) As String
    Dim theItemNumbers(20) As String
    Dim theHTSTotals(20) As Double, theHTSQuantity(20) As Double
    Dim theAVHTS(8) As String
    Dim theAVItemNumber(8) As String
    Dim theAVTotal(8) As Double, theTotal As Double, theFinalTotal As Double


    Dim rowCount As Integer, k As Integer, l As Integer, m As Integer, n As Integer
    Dim theInvalidSetList As String, isInvalid As Boolean
    Dim searchRange As Range, theRange As Range, theHTSRange As String, theItemNumberRange As String,      ↙
    theTotalChargeRange As Double
    Dim theAVRange As String, theErrorList As String
    Dim theRow As Integer, thePlaceRow As Integer, theSecondPlaceRow As Integer, theSetPlaceRow As Integer
    Dim totalCharge As Double, rowTwoCharge As Double, rowThreeCharge As Double, _
        rowFourCharge As Double, applicableRate As Double, theTempRow As Double

    rowCount = 3  '' initializes to thrid row of invoice sheet

    '' Main Loop -- Loops until the item number is blank
    '' Each pass increments rowCount
    While Not (Sheets("Invoice").Cells(rowCount, 2) = "") ''And Sheets("Invoice").Cells(rowCount + 2, 2) = ↙
    "" And Sheets("Invoice").Cells(rowCount + 3, 2) = "" And Sheets("Invoice").Cells(rowCount + 4, 2) = ""

        ''calls function to clean current item number
        Dim findItem As String
        findItem = Clean_Element(Sheets("Invoice").Cells(rowCount, 2))

        Dim htsCounter As Long ''counter through hts code list
        Dim duplicateCounter As Integer '' counter for multiple occurances of same item number
        duplicateCounter = 0

        ''searching through HTS Code List to find the current item number from Invoice (while rows not   ↙
    blank, and while recursive loop has not been started)
        ''once the recursion has started, we eliminate the need to look for the first occurance of finditem
        While ((Sheets("HTS Code List").Cells(htsCounter, 3) = "" And Sheets("HTS Code List").Cells      ↙
    (htsCounter, 4) = "" And Sheets("HTS Code List").Cells(htsCounter, 12) = "") And (duplicateCounter =    ↙
    0))
            ''if one is found, recursively checking through to see if there are duplicates of item number

            ''once we determine the exact methodology for rectifying the duplicates, it needs to be coded  ↙
    into this loop to prevent a recurse from happening when it is not needed (on supplier)
            If (Sheets("HTS Code List").Cells(htsCounter, 3) = findItem) Then
                ''if a cell matches the find item, initialize counter variable, and call recursion function
                If (duplicateCounter = 1 + Item_Duplicate_Check(findItem, htsCounter) < 2) Then
                    ''if there is only one occurance of the item number

                Else
                    ''if there is more than one occurance of the item number
                    '' recall that the recurssive function knows that any item numbers it finds are        ↙
    duplicates, and can immediately be thrown into the error spot
                    '' here we only need to take care of the first occurance of the duplicated value,      ↙
    nothing else


                End If

            End If
            htsCounter = htsCounter + 1 ''moving to the next row on hts code list
        End While
        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ↙
    ''''''''''''''''''''''''''''''''
        rowCount = rowCount + 1 ''move to the next row on invoice
    End While








    ''''------------------------------------OLD CODE--------------------------------------------------''''  ↙
    ''''''''''''''

    If (Not theRange Is Nothing Or findItem = "") Then
        theRow = theRange.Row  '' the row on the hts code list sheet
        theRange = Nothing
        '' checks to make sure sets have either same rate or percentages''
        If (Not Sheets("HTS Code List").Cells(theRow + 2, 4).Value = "" And _
        (Sheets("HTS Code List").Cells(theRow + 1, 3).Value = "" And Not Sheets("HTS Code List").Cells ↙
    (theRow + 1, 7).Value = "")) Then
            If ((Not Sheets("HTS Code List").Cells(theRow, 12).Value = Sheets("Invoice").Cells(rowCount, 5) ↙
     _
                And Sheets("HTS Code List").Cells(theRow + 1, 14).Value = "")) Then
                isInvalid = True
                GoTo Break7 '' easy way to skip all the code (not the best way however)
            End If
        End If
        isInvalid = False
        If (Not Sheets("HTS Code List").Cells(theRow, 12).Value = Sheets("Invoice").Cells(rowCount, 5))    ↙
    Then ' do the total rates match?
            If (Not i < 1) Then
                If Not IsNumeric(Sheets("HTS Code List").Cells(theRow, 12)) Then
                    applicableRate = Sheets("Invoice").Cells(rowCount, 5)
                Else
                    applicableRate = Sheets("Invoice").Cells(rowCount, 5) '' for all quotes that the rates ↙
    differ, use the invoice rate when not doing manual
                End If
            Else
                applicableRate = Sheets("Invoice").Cells(rowCount, 5)
            End If
        Else
            applicableRate = Sheets("HTS Code List").Cells(theRow, 12).Value
        End If
        totalCharge = applicableRate * Sheets("Invoice").Cells(rowCount, 4)  'row 1 charge
        '' Checking for a set, and then how large the set is
        If (Sheets("HTS Code List").Cells(theRow + 1, 3) = "" And Sheets("HTS Code List").Cells(theRow + 2, ↙
     3) = "") Then
            '' second row hts or description ?
            If ((Not Sheets("HTS Code List").Cells(theRow + 1, 4) = "" Or Not Sheets("HTS Code List").Cells ↙
    (theRow + 1, 7) = "")) Then
                If (Sheets("Invoice").Cells(rowCount, 4) > 0) Then '' if the quantity is a number greater  ↙
    than 0
                    If (Not applicableRate = Sheets("HTS Code List").Cells(theRow, 12).Value) Then '' if   ↙
    the rate is not the same as hts code list
                        If (Not Sheets("HTS Code List").Cells(theRow + 1, 14).Value = "") Then '' if       ↙
    percent exists
                            rowTwoCharge = applicableRate * Sheets("Invoice").Cells(rowCount, 4) * Sheets( ↙
    "HTS Code List").Cells(theRow + 1, 14) 'row 2 charge
                        Else
                            rowTwoCharge = 0
                            rowThreeCharge = 0
                            rowFourCharge = 0
                            GoTo theEnd2
                        End If
                    Else
                        rowTwoCharge = Sheets("HTS Code List").Cells(theRow + 1, 12) * Sheets("Invoice").  ↙
    Cells(rowCount, 4) 'row 2 charge
                    End If
                End If

                '' third row description or HTS ?
                If ((Not Sheets("HTS Code List").Cells(theRow + 2, 4) = "" Or Not Sheets("HTS Code List"). ↙
    Cells(theRow + 2, 7) = "")) Then
                    If (Sheets("Invoice").Cells(rowCount, 4) > 0) Then '' if the quantity is a number      ↙
    greater than 0
                        If (Not applicableRate = Sheets("HTS Code List").Cells(theRow, 12).Value) Then ''  ↙
    if the rate is not the same as hts code list
                            If (Not Sheets("HTS Code List").Cells(theRow + 2, 14).Value = "") Then '' if   ↙
    percent exists
                                rowThreeCharge = applicableRate * Sheets("Invoice").Cells(rowCount, 4) *   ↙
    Sheets("HTS Code List").Cells(theRow + 2, 14) 'row 3 charge
                            Else
```

```
                                GoTo theEnd2
                            End If
                        Else
                            rowThreeCharge = Sheets("HTS Code List").Cells(theRow + 2, 12) * Sheets(         ↙
    "Invoice").Cells(rowCount, 4) 'row 3 charge
                        End If

                        '' row 4 hts or description
                        If ((Not Sheets("HTS Code List").Cells(theRow + 3, 12) = "" Or Not Sheets("HTS Code ↙
     List").Cells(theRow + 3, 7) = "")) Then
                            If (applicableRate = Sheets("HTS Code List").Cells(theRow, 12).Value) Then   '' ↙
     if rates match
                                rowFourCharge = ((applicableRate * Sheets("Invoice").Cells(rowCount, 4)) -  ↙
    (rowTwoCharge + rowThreeCharge)) ''/ Sheets("Invoice").Cells(rowCount, 4) 'row 4 charge
                            Else
                                rowFourCharge = 0 '' default 45/55 % used
                            End If
                            ''Range("L3") = rowTwoCharge
                            ''Range("L4") = rowThreeCharge
                            ''Range("L5") = rowFourCharge

                            If (Not (rowTwoCharge + rowThreeCharge = 0 Or rowFourCharge = 0)) Then
                                theTempRow = rowTwoCharge
                                rowTwoCharge = rowTwoCharge + (rowTwoCharge / (rowTwoCharge +               ↙
    rowThreeCharge)) * rowFourCharge  '' final charges with box's on both items
                                rowThreeCharge = rowThreeCharge + (rowThreeCharge / (theTempRow +           ↙
    rowThreeCharge)) * rowFourCharge ''
                            End If
                            ''Range("M3") = rowTwoCharge
                            ''Range("M4") = rowThreeCharge
                            ''Range("M5") = rowFourCharge
                            ''Range("N5") = findItem
                        End If
                    End If
                End If
            Else
            End If
        End If
theEnd2:
        ''Range("M3") = rowTwoCharge
        '------------------Placement--------------------------'
        ''Range("M4") = rowThreeCharge
        ''Range("M5") = rowFourCharge
        ''Range("N5") = findItem
        '' find where the hts should be placed
        theHTSRange = Sheets("HTS Code List").Cells(theRow, 4)  '' the hts spots on the calculation tab
        For k = 0 To 20
            If (theHTSRange = theHTSCode(k)) Then '' look for element in the array,
                thePlaceRow = k
                GoTo Break3
            Else
                For l = 0 To 20
                    If ((theHTSCode(k)) = "") Then '' if the element isnt in array, find first  blank
                        thePlaceRow = k
                        GoTo Break3
                    End If
                Next l
            End If
        Next k
Break3:
        If (Sheets("HTS Code List").Cells(theRow, 8) = "Y") Then  '' if added value is needed
            theTotal = Sheets("HTS Code List").Cells(theRow, 10) * Sheets("Invoice").Cells(rowCount, 4)     ↙
     'raw mats cost
            theTotal = theTotal + Sheets("HTS Code List").Cells(theRow, 11) * Sheets("Invoice").Cells      ↙
    (rowCount, 4)     ' export freight value
            theAVRange = findItem
            For m = 0 To 8
                If ((theAVItemNumber(m)) = theAVRange) Then '' look for element in the array,
                    theSecondPlaceRow = m
                    GoTo Break4
                Else
                    For n = 0 To 8
                        If ((theAVItemNumber(m)) = "") Then '' if the element isnt in array, find first ↙
    blank
                            theSecondPlaceRow = m
                            GoTo Break4
                        End If
                    Next n
                End If
            Next m
Break4:
            theAVItemNumber(theSecondPlaceRow) = theAVRange
            theAVTotal(theSecondPlaceRow) = theAVTotal(theSecondPlaceRow) + theTotal
            theAVHTS(theSecondPlaceRow) = Sheets("HTS Code List").Cells(theRow, 4)

        End If


        theItemNumberRange = findItem
        theTotalChargeRange = Sheets("Invoice").Cells(rowCount, 5) * Sheets("Invoice").Cells(rowCount, 4)  ↙
    '' invoice unit price * invoice qty

        '' getting the correct hts set codes in the rows
        If (Sheets("HTS Code List").Cells(theRow + 1, 3) = "" And Sheets("HTS Code List").Cells(theRow + 2,↙
     3) = "") Then

            If ((Not Sheets("HTS Code List").Cells(theRow + 1, 4) = "" Or Not Sheets("HTS Code List").Cells↙
    (theRow + 1, 8) = "")) Then '' the second row is applicable, hence, a set
                ''theHTSRange = Sheets("HTS Code List").Cells(theRow, 4) & " / " & Sheets("HTS Code List").↙
    Cells(theRow + 1, 4)
                For m = 0 To 20
                    theHTSRange = Sheets("HTS Code List").Cells(theRow, 4) & " / " & Sheets("HTS Code List"↙
    ).Cells(theRow + 1, 4)
                    If (theHTSCode(m) = theHTSRange) Then '' look for element in the array,
                        theSetPlaceRow = m
                        GoTo Break5
                    Else
                        For n = 0 To 20
                            If (theHTSCode(m) = "") Then  '' if the element isnt in array, find first      ↙
    blank
                                theSetPlaceRow = m
                                GoTo Break5
                            End If
                        Next n
                    End If
                Next m
Break5:
                theHTSCode(theSetPlaceRow) = theHTSRange  '' puts the hts in correct matrix element
                If (Not theItemNumberRange = "") Then
                    theItemNumbers(theSetPlaceRow) = theItemNumbers(theSetPlaceRow) & theItemNumberRange & ↙
    ",  "
                End If
                If (rowTwoCharge = 0 And rowThreeCharge = 0 And rowFourCharge = 0) Then
                    theHTSTotals(theSetPlaceRow) = theHTSTotals(theSetPlaceRow) + totalCharge
                Else
                    theHTSTotals(theSetPlaceRow) = theHTSTotals(theSetPlaceRow) + rowTwoCharge
                End If
                theHTSQuantity(theSetPlaceRow) = theHTSQuantity(theSetPlaceRow) + Sheets("Invoice").Cells  ↙
    (rowCount, 4)

                If ((Not Sheets("HTS Code List").Cells(theRow + 2, 4) = "" Or Not Sheets("HTS Code List"). ↙
    Cells(theRow + 2, 8) = "")) Then '' the second row is applicable, hence, a set
```

```
                    For m = 0 To 20
                        theHTSRange = Sheets("HTS Code List").Cells(theRow, 4) & " / " & Sheets("HTS Code  ↙
    List").Cells(theRow + 2, 4)
                        If (theHTSCode(m) = theHTSRange) Then '' look for element in the array,
                            theSetPlaceRow = m
                            GoTo Break6
                        Else
                            For n = 0 To 20
                                If (theHTSCode(m) = "") Then  '' if the element isnt in array, find first  ↙
    blank
                                    theSetPlaceRow = m
                                    GoTo Break6
                                End If
                            Next n
                        End If
                    Next m
Break6:
                    theHTSCode(theSetPlaceRow) = theHTSRange  '' puts the hts in correct matrix element
                    If (Not theItemNumberRange = "") Then
                        theItemNumbers(theSetPlaceRow) = theItemNumbers(theSetPlaceRow) &                  ↙
    theItemNumberRange & ",  "
                    End If
                    If (rowTwoCharge = 0 And rowThreeCharge = 0 And rowFourCharge = 0) Then
                        theHTSTotals(theSetPlaceRow) = theHTSTotals(theSetPlaceRow)
                    Else
                        theHTSTotals(theSetPlaceRow) = theHTSTotals(theSetPlaceRow) + rowThreeCharge
                    End If
                    theHTSQuantity(theSetPlaceRow) = theHTSQuantity(theSetPlaceRow) + Sheets("Invoice").   ↙
    Cells(rowCount, 4)
                End If
                ''   Else '' when there is not a set, these are used
                ''      theHTSCode(thePlaceRow) = theHTSRange
                ''      If (Not theItemNumberRange = "") Then
                ''          theItemNumbers(thePlaceRow) = theItemNumbers(thePlaceRow) & theItemNumberRange ↙
    & ",  "
                ''      End If
                ''      theHTSTotals(thePlaceRow) = theHTSTotals(thePlaceRow) + theTotalChargeRange
                ''      theHTSQuantity(thePlaceRow) = theHTSQuantity(thePlaceRow) + Sheets("Invoice").Cells↙
    (rowCount, 4)
            End If
        Else '' when there is not a set, these are used
            theHTSCode(thePlaceRow) = theHTSRange
            If (Not theItemNumberRange = "") Then
                theItemNumbers(thePlaceRow) = theItemNumbers(thePlaceRow) & theItemNumberRange & ",  "
            End If
            theHTSTotals(thePlaceRow) = theHTSTotals(thePlaceRow) + theTotalChargeRange
            theHTSQuantity(thePlaceRow) = theHTSQuantity(thePlaceRow) + Sheets("Invoice").Cells(rowCount,  ↙
    4)
        End If
    Else
        theErrorList = theErrorList & findItem & ",  "
    End If
Break7:
    If (isInvalid) Then
        theInvalidSetList = theInvalidSetList & findItem & ",  "
    End If


    ''''------------------------------------END OLD CODE--------------------------------------------------  ↙
    ''''''''''''''''''

    Call Place_Values(theHTSCode, theItemNumbers, theHTSQuantity, theHTSTotals, theAVHTS, theAVItemNumber, ↙
    theAVTotal, theErrorList, theInvalidSetList, WorksheetFunction.Sum(theHTSTotals), WorksheetFunction.Sum↙
    (theAVTotal))
```

```
End Sub

Private Function Clean_Element(tempString As String) As String
    Dim p As Integer
    p = 1  ''initializes counter to first character in string

    '' moves through element, removing spaces until reaching end of element
    While p <= Len(tempString)
        If Mid(tempString, p, 1) <> " " Then ''if character is not a space, keep it
            Clean_Element = Clean_Element & Mid(tempString, p, 1)
        End If
    End While

    Return Clean_Element '' returns the space-free element
End Function

Private Sub ClearTotals() '' clears all data on worksheet
    Application.ScreenUpdating = False '' delays screen updates to speed up utility

    Range("$B$2:$H$22").Select() '' selects and clears contents of data elements on worksheet
    Selection.ClearContents()
    Range("$I$12:$K$22").Select()
    Selection.ClearContents()
    Range("$K$6").Select()
    Selection.ClearContents()
    Range("$C$23").Select()
    Selection.ClearContents()
    Range("$H$23").Select()
    Selection.ClearContents()
    Range("$B$2").Select()
End Sub

Private Sub Delete_Blank_HTS_Rows()
    '' code to get rid of blank lines on hts code list
    Dim r As Long '' in case integer isnt larg enough
    For r = Sheets("HTS Code List").Cells(Rows.Count, 1).End(xlUp).Row To 1 Step -1
        If Sheets("HTS Code List").Cells(r, 3) = "" And Sheets("HTS Code List").Cells(r, 4) = "" And Sheets ↙
    ("HTS Code List").Cells(r, 12) = "" Then Sheets("HTS Code List").Rows(r).Delete()
    Next r
End Sub

Private Sub Place_Values(htsCodes, itemNums, htsQty, htsTotals, avHTS, avItems, avTotals, errors, invalids, ↙
     htsFinal, avFinal)
    '' This sub takes all of the variables resulting from calculation, and puts them in the appropriate    ↙
    place on the worksheet for display
    Dim j As Integer
    For j = 2 To 22  '' putting all of the values into appropriate  cells from matricies
        Cells(j, 2) = htsCodes(j - 2)
        Cells(j, 3) = itemNums(j - 2)
        If (htsQty(j - 2) > 0) Then  '' only place values if its a value greater than 0
            Cells(j, 7) = htsQty(j - 2) '' lists quantity
        End If

        If (htsTotals(j - 2) > 0) Then  '' only place values if its a value greater than 0
            Cells(j, 8) = htsTotals(j - 2)
        End If
    Next j

    Dim o As Integer
    For o = 12 To 20
        Cells(o, 10) = avHTS(o - 12)
        Cells(o, 9) = avItems(o - 12)
        If (avTotals(o - 12) > 0) Then  '' only place values if its a value greater than 0
            Cells(o, 11) = avTotals(o - 12)
        End If
    Next o
    Range("C23").Value = theErrorList
    Range("H23").Value = theInvalidSetList
    Range("K6").Value = htsFinal '' place final total
    Range("K21").Value = avFinal '' place AV total

End Sub

Private Function GetOption(OpArray, theDefault, Title)
    Dim TempForm  'As VBComponent
    Dim NewOptionButton As Msforms.OptionButton
    Dim NewCommandButton1 As Msforms.CommandButton
    Dim NewCommandButton2 As Msforms.CommandButton
    Dim TextLocation As Integer
    Dim X As Integer, i As Integer, TopPos As Integer
    Dim MaxWidth As Long
    Dim WasVisible As Boolean

    '   Hide VBE window to prevent screen flashing
    Application.VBE.MainWindow.Visible = False

    '   Create the UserForm
    TempForm = ThisWorkbook.VBProject.VBComponents.Add(3)
    TempForm.Properties("Width") = 800

    '   Add the OptionButtons
    TopPos = 4
    MaxWidth = 0 'Stores width of widest OptionButton
    For i = LBound(OpArray) To UBound(OpArray)
        NewOptionButton = TempForm.Designer.Controls.Add("forms.OptionButton.1")
        With NewOptionButton
            .Width = 800
            .Caption = OpArray(i)
            .Height = 15
            .Left = 8
            .Top = TopPos
            .Tag = i
            .AutoSize = True
            If theDefault = i Then .Value = True
            If .Width > MaxWidth Then MaxWidth = .Width
        End With
        TopPos = TopPos + 15
    Next i

    '   Add the Cancel button
    NewCommandButton1 = TempForm.Designer.Controls.Add("forms.CommandButton.1")
    With NewCommandButton1
        .Caption = "Cancel"
        .Height = 18
        .Width = 44
        .Left = MaxWidth + 12
        .Top = 6
    End With

    '   Add the OK button
    NewCommandButton2 = TempForm.Designer.Controls.Add("forms.CommandButton.1")
    With NewCommandButton2
        .Caption = "OK"
        .Height = 18
        .Width = 44
        .Left = MaxWidth + 12
        .Top = 28
    End With

    '   Add event-hander subs for the CommandButtons
    With TempForm.CodeModule
```

```
        X = .CountOfLines
        .InsertLines(X + 1, "Sub CommandButton1_Click()")
        .InsertLines(X + 2, "  GETOPTION_RET_VAL=False")
        .InsertLines(X + 3, "  Unload Me")
        .InsertLines(X + 4, "End Sub")

        .InsertLines(X + 5, "Sub CommandButton2_Click()")
        .InsertLines(X + 6, "  Dim ctl")
        .InsertLines(X + 7, "  GETOPTION_RET_VAL = False")
        .InsertLines(X + 8, "  For Each ctl In Me.Controls")
        .InsertLines(X + 9, "    If ctl.Tag <> """" Then If ctl Then GETOPTION_RET_VAL = ctl.Tag")
        .InsertLines(X + 10, "  Next ctl")
        .InsertLines(X + 11, "  Unload Me")
        .InsertLines(X + 12, "End Sub")
    End With

    '   Adjust the form
    With TempForm
        .Properties("Caption") = Title
        .Properties("Width") = NewCommandButton1.Left + NewCommandButton1.Width + 10
        If .Properties("Width") < 160 Then
            .Properties("Width") = 160
            NewCommandButton1.Left = 106
            NewCommandButton2.Left = 106
        End If
        .Properties("Height") = TopPos + 24
    End With

    '   Show the form
    VBA.UserForms.Add(TempForm.Name).Show()

    '   Delete the form
    ThisWorkbook.VBProject.VBComponents.Remove VBComponent:=TempForm

    '   Pass the selected option back to the calling procedure
    GetOption = GETOPTION_RET_VAL
End Function
```