```
'************************************************
'PROGRAM NAME:          CIS-14 Final Project
'                       "Capital Gains Calculator for Equity Transactions"
'
'CREATION DATE:         02-22-2004
'
'REV DATE:              XX-XX-XXXX
'REV LEVEL:             X1
'SOFTWARE VERSION:      Visual BASIC Version 6.0
'                       Working Model Edition
'
'STUDENT PROGRAMMER:    RON SELMAN, email: ron-selman@comcast.net
'
'INSTRUCTOR:            MR. JOHN CRANE
'
'PROGRAM SUMMARY:       This program calculates Capital Gains Tax for
'                       typical brokerage equity trading.  Elapsed time periods
'                       are entered across year boundries to show tax due or
'                       tax credits for the period.
'                       The program cannot support long term capital gains....
'                       The program finds gains and losses in trades for issues
'                       in the time period.
'
'                       How well can this be broken down into illustrations?
'
'                       The program requests a filespec for a *.csv file of
'                       transactions.  Then determines if the format is correct.
'
'                       It enters the file into an array in order of entry.
'                       It then parses out the entries into groups of the
'                       individual issues, again in the same order as entry.
'                       The program requests a time period to process for
'                       Capital Gains.
'
'*****************************************************************
'VARIABLE DECLARATION SECTION----------------------------------------

Dim Array_in(0 To 500) As String
Dim Array_out(0 To 500) As String
Dim T_array(0 To 500) As String
Dim C_array(0 To 500) As String
Dim U_array(0 To 500) As String

'Variables for miscellaneous uses
Dim val_row As Integer, k As Integer
Dim date_start As String, date_end As String
Dim InputFileSpec As String
Dim OutputFileSpec As String
Dim help_file As String, array_str As String, array_str1 As String, process_log As String

Const ShortTermTaxRate = 0.391 'Tax Rate for short term investments.

Private Sub cmdReprocessArray_Click()
'Take array string from the OutputArray sub routine and reloaded it into an array for reprocessing.
    Dim i As Integer, j As Integer, tmp_str As String, tmp_str1 As String, CRLF1 As String
    CRLF1 = Chr(13) & Chr(10)
    tmp_str = array_str

    Do While (Chr(13) = Left(tmp_str, 1) Or Chr(10) = Left(tmp_str, 1))
        tmp_str = Mid(tmp_str, 2)
    Loop

    Array_in(0) = ""
    Do
        j = InStr(tmp_str, CRLF1) - 1   'Find next CRLF
        If j > 0 Then
            tmp_str1 = Left(tmp_str, j)     'Get the line you want
            Array_in(i) = tmp_str1          'Place in the array
            tmp_str = Mid(tmp_str, j + 3)     'Get all the string past this CRLF for later
        End If
```

```
    Loop Until (10 > Len(tmp_str))

    Call OutputArray(Array_in(), 0, val_row, array_str, "Temp Sort Report File")

    Call SortArray(Array_in(), Array_out(), val_row)
    Call OutputArray(Array_out(), 0, val_row, array_str1, "Order Issues Report File")
    cmdSpreadArray.Visible = False
    cmdOutputFileSpec.Visible = True

End Sub

Public Sub Form_Load()

End Sub

Private Sub Form_Activate()
    InputFileSpec = ""
    OutputFileSpec = "A:\CIS-14A FILES\Final Project Archive\Final Project\CapGains Report to 02-26-04.csv"
    date_start = "": date_end = ""

    cmdRunInSteps.Visible = False
    cmdCompleteRun.Visible = False
    cmdOutputFileSpec.Visible = False
    cmdPrintReport.Visible = False
    cmdSpreadArray.Visible = False
    cmdCalcBefore.Visible = False
    cmdCalcDuring.Visible = False
    cmdSumTax.Visible = False
    cmdInputFilespec.Visible = True
    cmdHelpClose.Visible = False
    cmdHelp.Visible = True
    cmdReprocessArray.Visible = False
    lblCopyright.Visible = True
    lblCopyright.Caption = Chr(169) & "2004 Ron Selman"
    lblTitlePicBox.Font.Size = 19
    lblTitlePicBox.Caption = "Program Status Log"
    picFileSpec.AutoRedraw = True
    picFileSpec.Cls
    picFileSpec.Print "The program is starting."
    process_log = "The program is starting." & Chr(13) & Chr(10)

End Sub

Public Sub BuildArray(in_array() As String, v_row As Integer, f_name As String)
    Dim i As Integer, s As String   'Dim temp variables
    i = 0: s = ""                   'Init temp variables
    Open f_name For Input As #1     'Open *.csv file first time
    Do While Not EOF(1)             'Find out how many lines are in *.csv file.
        Line Input #1, s            'Toss this string.
        i = i + 1                   'Accumulate count of the number of lines.
    Loop

    v_row = i                       'assign # of rows for an index for use in array
    Close #1                        'Close file so it can be opened at start.
'    in_array(0) = Str(v_row)        'Place the number of lines in the first array element. ED NOTE: Remove line
    Open f_name For Input As #1     'Open *.csv file second time
    For i = 1 To v_row Step 1       'Place each line from file into the array elements.
        Line Input #1, in_array(i)
    Next i
    If Not (EOF(1)) Then            'Check end of file was found, process error if was not EOF.
        picFileSpec.Print "File: " & f_name & " End of File not Found,  File Error!"
    Else
        picFileSpec.Print "Opening input filename: " & f_name
    End If
    If in_array(1) <> "Symbol,Quantity,Price,Action,TradeDate,Principal,Commission,SECFees,NetAmount,AccountType,Memo" Then
        picFileSpec.Print "Input is not proper Equities Trade *.csv file!"  'Process error if found
    Else
        picFileSpec.Print "Input file: " & f_name & " is a proper Equities Trade *.csv file." 'Proper file was found.
    End If
    Close #1        'Close the file, data is now in array.
    cmdCompleteRun.Visible = True
```

```
    cmdRunInSteps.Visible = True
End Sub

Public Sub cmdCompleteRun_Click()
    Call cmdHelpClose_Click
    cmdRunInSteps.Visible = False
    cmdCompleteRun.Visible = False
    date_start = InputBox("Please, enter a date: [E.G. mm/dd/yyy]", "Enter the first date to include in the calculations")
    date_end = InputBox("Please, enter a date: [E.G. mm/dd/yyy]", "Enter the last date to include in the calculations")

    Call SortArray(Array_in(), Array_out(), val_row)

    Call SpreadArray(Array_out(), T_array(), val_row)

    cmdOutputFileSpec.Visible = True

    Call CalcBefore(T_array(), 1, val_row, date_start)

    Call CalcDuring(T_array(), 1, val_row, date_start, date_end)

    Call copyArrayToArray(T_array(), U_array(), 5, 0, val_row)

    Call SumTax(U_array(), 6, val_row)

    Call OutputArray(U_array(), 0, val_row, array_str, "Report File")

End Sub

Private Sub cmdRunInSteps_Click()
 'The raw *.csv file has been loaded
 'An initial array has been loaded
 'The array has been sorted for issues
 'A space has been placed between issues and the number of issues inserted.

    Call cmdHelpClose_Click
    cmdCompleteRun.Visible = False

    date_start = InputBox("Please, enter a date: [E.G. mm/dd/yyy]", "Enter the first date to include in the calculations")
    date_end = InputBox("Please, enter a date: [E.G. mm/dd/yyy]", "Enter the last date to include in the calculations")

    Call SortArray(Array_in(), Array_out(), val_row)
    Call OutputArray(Array_out(), 0, val_row, array_str, "Order Issues Report File")

    cmdOutputFileSpec.Visible = True

    cmdSpreadArray.Visible = True
    cmdRunInSteps.Visible = False
    cmdReprocessArray.Visible = True
End Sub

Private Sub cmdSpreadArray_Click()

    Call SpreadArray(Array_out(), T_array(), val_row)
    Call OutputArray(T_array(), 0, val_row, array_str, "Spread Array Report File")
    cmdOutputFileSpec.Visible = True
    cmdSpreadArray.Visible = False
    cmdCalcBefore.Visible = True
    cmdReprocessArray.Visible = False
End Sub

Private Sub cmdCalcBefore_Click()
    cmdSpreadArray.Visible = False
    Call CalcBefore(T_array(), 1, val_row, date_start)
    Call OutputArray(T_array(), 0, val_row, array_str, "Calculate Before Report File")
    cmdOutputFileSpec.Visible = True
    cmdCalcBefore.Visible = False
    cmdCalcDuring.Visible = True
    cmdReprocessArray.Visible = False
End Sub
```

```
Private Sub cmdCalcDuring_Click()
    cmdCalcBefore.Visible = False
    Call CalcDuring(T_array(), 1, val_row, date_start, date_end)
    Call OutputArray(T_array(), 0, val_row, array_str, "Calculate During Report File")
    Call copyArrayToArray(T_array(), U_array(), 5, 0, val_row)
    cmdOutputFileSpec.Visible = True
    cmdCalcDuring.Visible = False
    cmdSumTax.Visible = True
    cmdReprocessArray.Visible = False
End Sub
Private Sub cmdSumTax_Click()
    cmdCalcDuring.Visible = False
    Call SumTax(U_array(), 6, val_row)
    Call OutputArray(U_array(), 0, val_row, array_str, "Calculate Tax Report File")
    cmdOutputFileSpec.Visible = True
    cmdSumTax.Visible = False
    cmdReprocessArray.Visible = False
End Sub

Public Sub SumTax(in_array() As String, start_for As Integer, end_for As Integer)
'The array has been processed for transactions before the date range and during the date range.
'Info from after the date range was never calculated as it is not needed.
'All relevant information to calculate such things as loss, profit, and tax due is in the
'comment string for each issue.  Open issues and bankrupt issues that never closed still remain.
'These remnants can be cleared if desired and the SumTax() subroutine or equivalent will be run again.

    Dim i As Integer, j As Integer, k As Integer, l As Integer, m As Integer
    Dim tmp_str As String
    Dim before_shares As Single, open_remaining_shares As Single, p_total_profit_loss As Currency, total_profit_loss As Currency
    Dim taxable_amount As Currency, remnant_shares As Single, acc_remnant_shares As Single, currency_var As Currency
    before_shares = 0: open_remaining_shares = 0: p_total_profit_loss = 0: total_profit_loss = 0
    taxable_amount = 0: remnant_shares = 0: acc_remnant_shares = 0

'We look at the comment string elements of the array only.
'The result is placed at the beginning of the array.

    i = start_for
    Do
        j = xtract_num_issues(in_array(), i)

        before_shares = xtractNum(in_array(), i, "before_shares=")
        open_remaining_shares = xtractNum(in_array(), i, "remaining_shares=")
        p_total_profit_loss = xtractNum(in_array(), i, "before_cost=")
        total_profit_loss = xtractNum(in_array(), i, "profit_or_loss=")

        'There are no shares remaining from before the date period.
        If before_shares = 0 Then
            taxable_amount = taxable_amount + total_profit_loss 'Tally toward the date period taxable_amount.
            in_array(i) = in_array(i) & ", Taxable= " & total_profit_loss
        End If

        'There are shares remaining from before the time period.
        'If these shares are unsold during the time period we don't care.
        'If they are sold during the date period they must be taxed.
        If before_shares > 0 Then
            If (before_shares + open_remaining_shares = 0) Then
                taxable_amount = taxable_amount + total_profit_loss + p_total_profit_loss
                in_array(i) = in_array(i) & ", Taxable= " & total_profit_loss + p_total_profit_loss
            End If

        'If more shares have been sold (or buy short) than found in the account, then the file has not been reconciled.
            If (before_shares + open_remaining_shares < 0) Then
                remnant_shares = before_shares + open_remaining_shares
                in_array(i) = in_array(i) & ", Remnants= " & remnant_shares     'Mark the issue.
                acc_remnant_shares = acc_remnant_shares + Abs(remnant_shares)   'Tally the number of remnants.
            End If
        End If

        'If there are shares from before the time period, but only some of them are sold during the time period.
```

```
        i = i + j + 1

    Loop Until (i >= end_for)
'Each issue has had the remaining shares and costs calculated and placed in the comment line for later use.
'The sum total of profit_loss, and open shares is place in the first line of the array for view or processing.
    in_array(0) = "Capital Gains Tax Calculator for Personal Equity Accounts Report"
    in_array(1) = "Path and filename of input file is: " & InputFileSpec & ", Today's Date: " & Date & ", Time of Day: " & Time
    in_array(2) = "Tax calculation time period is: Start Date= " & date_start & " to End Date= " & date_end
    in_array(3) = "File processing error information: Open Remaining Shares= " & Str(open_remaining_shares) & ", Accumulated Remnants Shares= " & Str(acc_remnant_shares)
    in_array(4) = "Taxable Amount= " & FormatCurrency(taxable_amount, 2) & ", Fed Tax Due @ " & FormatPercent(ShortTermTaxRate, 1) & " is " & FormatCurrency(ShortTermTaxRate * taxable_amount, 2)
End Sub

Public Sub CalcBefore(in_array() As String, start_for As Integer, end_for As Integer, d_lo As String)
'Looks through the entire array up to "d_lo" value to calculate cap gains tax credit due.
'Places result in the notes row now marked Calc'ed 'name of equity', 'number of occurrances' for each group of issues.
Dim i As Integer, j As Integer, k As Integer, l As Integer, m As Integer
Dim tmp_str As String, pre_remaining_shares As Single, pre_profit_loss As Single
'You must start at the beginning of each issue section to do the average cost basis processing correctly.

i = start_for
Do

    j = Val(Mid(in_array(i), InStr(in_array(i), ",") + 1))
    k = i + j
    For l = k To (i + 1) Step -1

        If ((InStr(in_array(l), "Buy") <> 0) And (0 = InStr(in_array(l), "Short"))) Then
            If (beforeDate(in_array(), l, d_lo)) Then
                pre_remaining_shares = pre_remaining_shares + xtract_num_shares(in_array(), l)
                pre_profit_loss = pre_profit_loss - xtract_principle(in_array(), l)
            End If
        End If
        If ((InStr(in_array(l), "Sell") <> 0) And (0 = InStr(in_array(l), "Short"))) Then
            If (beforeDate(in_array(), l, d_lo)) Then
                pre_remaining_shares = pre_remaining_shares - xtract_num_shares(in_array(), l)
                pre_profit_loss = pre_profit_loss + xtract_principle(in_array(), l)
            End If
        End If

         If ((InStr(in_array(l), "Sell") <> 0) And (0 <> InStr(in_array(l), "Short"))) Then
            If (beforeDate(in_array(), l, d_lo)) Then
                pre_remaining_shares = pre_remaining_shares + xtract_num_shares(in_array(), l)
                pre_profit_loss = pre_profit_loss - xtract_principle(in_array(), l)
            End If
        End If

        If ((InStr(in_array(l), "Buy") <> 0) And (0 <> InStr(in_array(l), "Short"))) Then
           If (beforeDate(in_array(), l, d_lo)) Then
                pre_remaining_shares = pre_remaining_shares - xtract_num_shares(in_array(), l)
                pre_profit_loss = pre_profit_loss + xtract_principle(in_array(), l)
            End If
        End If

    Next l
    in_array(l) = in_array(l) & ", before_shares= " & Str(pre_remaining_shares) & ", before_cost= " & Str(pre_profit_loss)
    i = i + j + 1
    pre_profit_loss = 0: pre_remaining_shares = 0

Loop Until (i >= end_for)
'Each issue has had the remaining shares and costs calculated and placed in the comment line for later use.
End Sub

Public Sub CalcDuring(in_array() As String, start_for As Integer, end_for As Integer, d_lo As String, d_hi As String)

Dim i As Integer, j As Integer, k As Integer, l As Integer, m As Integer
Dim tmp_str As String, remaining_shares As Single, profit_loss As Single
'You must start at the beginning of each issue section to do the processing correctly.
```

```
i = start_for
Do
    j = xtract_num_issues(in_array(), i)
    k = i + j
    For l = k To (i + 1) Step -1
        '
        If ((InStr(in_array(l), "Buy") <> 0) And (0 = InStr(in_array(l), "Short"))) Then
            If (Not beforeDate(in_array(), l, d_lo)) And (Not afterDate(in_array(), l, d_hi)) Then
                remaining_shares = remaining_shares + xtract_num_shares(in_array(), l)
                profit_loss = profit_loss - xtract_principle(in_array(), l)
            End If
        End If

        If ((InStr(in_array(l), "Sell") <> 0) And (0 = InStr(in_array(l), "Short"))) Then
            If (Not beforeDate(in_array(), l, d_lo)) And (Not afterDate(in_array(), l, d_hi)) Then
                remaining_shares = remaining_shares - xtract_num_shares(in_array(), l)
                profit_loss = profit_loss + xtract_principle(in_array(), l)
            End If
        End If

         If ((InStr(in_array(l), "Sell") <> 0) And (0 <> InStr(in_array(l), "Short"))) Then
            If (Not beforeDate(in_array(), l, d_lo)) And (Not afterDate(in_array(), l, d_hi)) Then
                remaining_shares = remaining_shares + xtract_num_shares(in_array(), l)
                profit_loss = profit_loss - xtract_principle(in_array(), l)
            End If
        End If

        If ((InStr(in_array(l), "Buy") <> 0) And (0 <> InStr(in_array(l), "Short"))) Then
           If (Not beforeDate(in_array(), l, d_lo)) And (Not afterDate(in_array(), l, d_hi)) Then
                remaining_shares = remaining_shares - xtract_num_shares(in_array(), l)
                profit_loss = profit_loss + xtract_principle(in_array(), l)
            End If
        End If

    Next l
    in_array(l) = in_array(l) & ", remaining_shares= " & Str(remaining_shares) & ", profit_or_loss= " & Str(profit_loss)
    i = i + j + 1
    profit_loss = 0: remaining_shares = 0

Loop Until (i >= end_for)
'Each issue has had the remaining shares and costs calculated and placed in the comment line for later use.

End Sub

Public Sub SpreadArray(in_array() As String, out_array() As String, v_row As Integer)
                                        'This Sub places an index array element between groups of
                                        'like equity issues in a new array. The index element has
                                        '"Calc'ed: 'issue name', 'number of appearances'"
                                        'Place the index element in front of the first entry for new array.
                                        'Read the first name, and check for additional same name issues.
                                        'If issue name is same then copy to new array.
                                        'When new name appears, then place a index array element in new array.
    Dim i As Integer, j As Integer, k As Integer, tmp_str As String
    i = 2: j = 2: k = 0:  tmp_str = "":  out_array(j) = ""

    Do
        k = 1
        out_array(j) = in_array(i)
        tmp_str = Left(in_array(i), InStr(in_array(i), ","))
        Do While ((tmp_str = Left(in_array(i + 1), InStr(in_array(i + 1), ","))) And (i < v_row)) 'If the present and the next are the same, move
            i = i + 1: j = j + 1: k = k + 1
            out_array(j) = in_array(i)
        Loop
        out_array(j - k) = "Calc'ed: " & Mid(tmp_str, 7) & Str(k)
        i = i + 1: j = j + 2
    Loop Until i > v_row
    v_row = j - 1
'    out_array(0) = v_row                  'Place size of array in the first element.
End Sub

Public Sub SortArray(in_array() As String, out_array() As String, v_row As Integer)
```

```
    'Separates the issues into groups per their appearance in the original file.
    Dim i As Integer, j As Integer, k As Integer, l_str As String   'Dim temp variables
    i = 0: k = 1: l_str = ""                                        'Init temp variables

    For i = 2 To v_row
        'Check the in_array() entry is an equity and place it in the out_array().
        If 1 = InStr(in_array(i), "EQUITY") Then
            k = k + 1
            out_array(k) = in_array(i)
            l_str = Left(in_array(i), InStr(in_array(i), ","))
            in_array(i) = "Transfered"
        ElseIf in_array(i) <> "Transfered" Then
            picFileSpec.Print "Problem with entry!  This was not an equity." & ", i =" & Str(i) & ", j = " & Str(j)
            picFileSpec.Print in_array(i)
            'Expand the error processing here.  Make user prompt message.
        End If
        'Now search the rest of the in_array() for additional entries.  Move and mark Transfered
        For j = 3 To v_row
            If (l_str = Left(in_array(j), InStr(in_array(j), ","))) Then
                k = k + 1
                out_array(k) = in_array(j)
                in_array(j) = "Transfered"
            End If
        Next j
    Next i
'    out_array(0) = k
    picFileSpec.Print "The array is sorted"
End Sub

Public Sub OutputArrayReport(Array_var() As String, start_for As Integer, end_for As Integer, lo_d As String, hi_d As String)
    Dim k As Integer, tmp_str As String, ArrayAsString As String
    ArrayAsString = ""
    For k = start_for To end_for
        tmp_str = Array_var(k)
        ArrayAsString = ArrayAsString & Chr(13) & Chr(10) & tmp_str
    Next k
    txtTextBox.Text = ArrayAsString
End Sub

Public Sub OutputArray(Array_var() As String, start_for As Integer, end_for As Integer, ret_str As String, lbl_caption)
    Dim k As Integer, tmp_str As String, ArrayAsString As String
    ArrayAsString = ""
    lblTextBox.Font.Size = 19
    lblTextBox.Caption = lbl_caption
    For k = start_for To end_for
        tmp_str = Array_var(k)
        ArrayAsString = ArrayAsString & Chr(13) & Chr(10) & tmp_str
    Next k
    ArrayAsString = ArrayAsString & Chr(13) & Chr(10)
    txtTextBox.Font.Size = 8
    txtTextBox.Text = ArrayAsString
    ret_str = ArrayAsString
End Sub

Private Sub cmdExit_Click()
    End
End Sub

Public Sub PrintArray(Array_prt() As String, start_for As Integer, end_for As Integer)
    Dim k As Integer, tmp_str As String, ArrayAsString As String
    Printer.Font.Size = 7
    ArrayAsString = ""
    For k = start_for To end_for
        tmp_str = Array_prt(k)
        ArrayAsString = ArrayAsString & Chr(13) & Chr(10) & tmp_str
    Next k
    Printer.Print ArrayAsString
    Printer.EndDoc
End Sub

Public Function beforeDate(Array_var() As String, ele_var As Integer, before_date As String) As Boolean
```

```
    Dim tmp_date As String, f_comma As Integer, s_comma As Integer
    before_date = DateValue(before_date)
    If (0 <> InStr(Array_var(ele_var), "Buy")) Then 'String position for Buy is different then Sell
        f_comma = InStr(Array_var(ele_var), "Buy") + 4
        tmp_date = Mid(Array_var(ele_var), f_comma)
        s_comma = InStr(tmp_date, ",") - 1
        tmp_date = DateValue(Mid(tmp_date, 1, s_comma))
    End If
    If (0 <> InStr(Array_var(ele_var), "Sell")) Then 'String position for Sell is different then Buy
        f_comma = InStr(Array_var(ele_var), "Sell") + 5
        tmp_date = Mid(Array_var(ele_var), f_comma)
        s_comma = InStr(tmp_date, ",") - 1
        tmp_date = DateValue(Mid(tmp_date, 1, s_comma))
    End If

    If Year(tmp_date) < Year(before_date) Then
        beforeDate = True
    End If
    If Year(tmp_date) > Year(before_date) Then
        beforeDate = False
    End If
    If (Year(tmp_date) = Year(before_date)) And (Month(tmp_date) < Month(before_date)) Then
        beforeDate = True
    End If
    If (Year(tmp_date) = Year(before_date)) And (Month(tmp_date) > Month(before_date)) Then
        beforeDate = False
    End If
    If (Year(tmp_date) = Year(before_date)) And (Month(tmp_date) = Month(before_date)) And (Day(tmp_date) < Day(before_date)) Then
        beforeDate = True
    End If
    If (Year(tmp_date) = Year(before_date)) And (Month(tmp_date) = Month(before_date)) And (Day(tmp_date) >= Day(before_date)) Then
        beforeDate = False
    End If
End Function
```

```
Public Function afterDate(Array_var() As String, ele_var As Integer, after_date As String) As Boolean
    Dim tmp_date As String, f_comma As Integer, s_comma As Integer
    after_date = DateValue(after_date)
    If (0 <> InStr(Array_var(ele_var), "Buy")) Then 'String position for Buy is different then Sell
        f_comma = InStr(Array_var(ele_var), "Buy") + 4
        tmp_date = Mid(Array_var(ele_var), f_comma)
        s_comma = InStr(tmp_date, ",") - 1
        tmp_date = Mid(tmp_date, 1, s_comma)
    End If
    If (0 <> InStr(Array_var(ele_var), "Sell")) Then 'String position for Sell is different then Buy
        f_comma = InStr(Array_var(ele_var), "Sell") + 5
        tmp_date = Mid(Array_var(ele_var), f_comma)
        s_comma = InStr(tmp_date, ",") - 1
        tmp_date = DateValue(Mid(tmp_date, 1, s_comma))
    End If

    If Year(tmp_date) > Year(after_date) Then
        afterDate = True
    End If
    If Year(tmp_date) < Year(after_date) Then
        afterDate = False
    End If
    If (Year(tmp_date) = Year(after_date)) And (Month(tmp_date) > Month(after_date)) Then
        afterDate = True
    End If
    If (Year(tmp_date) = Year(after_date)) And (Month(tmp_date) < Month(after_date)) Then
        afterDate = False
    End If
    If (Year(tmp_date) = Year(after_date)) And (Month(tmp_date) = Month(after_date)) And (Day(tmp_date) > Day(after_date)) Then
        afterDate = True
    End If
    If (Year(tmp_date) = Year(after_date)) And (Month(tmp_date) = Month(after_date)) And (Day(tmp_date) <= Day(after_date)) Then
        afterDate = False
    End If
End Function
```

```
Public Function xtract_num_issues(Array_var() As String, ele_var As Integer) As Integer
    Dim f_comma As Single, s_commas As Single, tmp_str1 As String
    f_comma = InStr(Array_var(ele_var), ",")
    tmp_str1 = Mid(Array_var(ele_var), f_comma + 1) 'Get quantity of issue
    s_comma = InStr(tmp_str1, ",")
    xtract_num_issues = Val(Mid(tmp_str1, 1, (s_comma - 1)))

End Function

Public Function xtract_principle(Array_var() As String, ele_var As Integer) As Single
    Dim f_comma As Single, s_commas As Single, tmp_str1 As String
    f_comma = InStr(Array_var(ele_var), ",")
    tmp_str1 = Mid(Array_var(ele_var), f_comma + 1) 'Get quantity of issue

    f_comma = InStr(tmp_str1, ",")
    tmp_str1 = Mid(tmp_str1, f_comma + 1) 'Get quantity of issue

    f_comma = InStr(tmp_str1, ",")
    tmp_str1 = Mid(tmp_str1, f_comma + 1) 'Get quantity of issue

    f_comma = InStr(tmp_str1, ",")
    tmp_str1 = Mid(tmp_str1, f_comma + 1) 'Get quantity of issue

    f_comma = InStr(tmp_str1, ",")
    tmp_str1 = Mid(tmp_str1, f_comma + 1) 'Get quantity of issue

    f_comma = InStr(tmp_str1, ",")
    xtract_principle = Val(Mid(tmp_str1, 1, (f_comma - 1)))

End Function

Public Function xtract_num_shares(Array_var() As String, ele_var As Integer) As Single
    Dim f_comma As Single, s_commas As Single, tmp_str1 As String
    f_comma = InStr(Array_var(ele_var), ",")
    tmp_str1 = Mid(Array_var(ele_var), f_comma + 1) 'Get quantity of issue
    s_comma = InStr(tmp_str1, ",")
    xtract_num_shares = Val(Mid(tmp_str1, 1, (s_comma - 1)))
End Function

Public Function xtractNum(Array_var() As String, ele_var As Integer, key_string As String)
    Dim f_point As Single, s_point As Single, tmp_str1 As String
    f_point = InStr(Array_var(ele_var), key_string) + Len(key_string)
    tmp_str1 = Mid(Array_var(ele_var), f_point + 1) 'Get quantity of issue
    s_point = InStr(tmp_str1, ",")
    If s_point = 0 Then: s_point = Len(tmp_str1) + 1
    xtractNum = Val(Mid(tmp_str1, 1, (s_point - 1)))
End Function

Public Function xtract_o_num_shares(Array_var() As String, ele_var As Integer) As Integer
    Dim f_comma As Single, s_commas As Single, tmp_str1 As String
    f_comma = InStr(Array_var(ele_var), "=")
    tmp_str1 = Mid(Array_var(ele_var), f_comma + 1) 'Get quantity of issue
    s_comma = InStr(tmp_str1, ",")
    xtract_o_num_shares = Val(Mid(tmp_str1, 1, (s_comma - 1)))
End Function

Public Function xtract_t_profit_loss(Array_var() As String, ele_var As Integer) As Integer
    Dim f_comma As Single, s_commas As Single, tmp_str1 As String
    f_comma = InStr(Array_var(ele_var), "=")
    tmp_str1 = Mid(Array_var(ele_var), f_comma + 1) 'Get quantity of issue
    s_comma = InStr(tmp_str1, "=")
    xtract_t_profit = Val(Mid(tmp_str1, 1, (s_comma - 1)))
End Function

Public Sub initArrayToNull(Array_var_in() As String, start_var As Integer, end_var As Integer)
    Dim i As Integer
    For i = start_var To end_var  'Copy contents so original array is preserved.
        Array_var_in(i) = ""
    Next i
End Sub
```

```
Public Sub copyArrayToArray(Array_var_in() As String, Array_var_out() As String, offset_var As Integer, start_var As Integer, end_var As Integer)
    Dim i As Integer
    For i = start_var To end_var  'Copy contents so original array is preserved.
        Array_var_out(i + offset_var) = Array_var_in(i)
    Next i
    end_var = end_var + offset_var 'Changes end_var valut to include the offset value.
End Sub

Private Sub cmdAbout_Click()
    picFileSpec.Cls
    picFileSpec.Print "Capital Gains Tax Calculator for Personal Equity Accounts"
    picFileSpec.Print "Version 1.0, Rev X1"
    picFileSpec.Print "Release Date: 03/25/2004"
    picFileSpec.Print "Written by Ron Selman, " & Chr(169) & "2004 Ron Selman"
    picFileSpec.Print "This program was a final project for CIS-14A."
End Sub

Private Sub cmdReset_Click()
array_str = ""
process_log = ""
txtTextBox.Text = ""

Call initArrayToNull(Array_in(), 0, 500)
Call initArrayToNull(Array_out(), 0, 500)
Call initArrayToNull(T_array(), 0, 500)
Call initArrayToNull(C_array(), 0, 500)
Call initArrayToNull(U_array(), 0, 500)
Call Form_Activate

End Sub


Private Sub cmdHelp_Click()
lblCopyright.Visible = False
cmdHelp.Visible = False
cmdHelpClose.Visible = True
lblTextBox.Font.Size = 19
lblTextBox.Caption = "Help Screen"
txtTextBox.Font.Size = 11
help_file = "                    Capital Gains Tax Calculator for Personal Brokerage Accounts." & Chr(13) & Chr(10) & Chr(13) & Chr(10) & _
            "The Start Screen configures as program starts.  It will request an input file with the button:" & Chr(13) & Chr(10) & _
            """Enter an Input File Name"", you must LClick this button, then select a ""*.csv"" file with the proper format." & Chr(13) & Chr(10) & _
            "E.G.: ""Cleaned_Sample.csv"", or ""Personal_Account1_not_reconciled.csv""" & Chr(13) & Chr(10) & Chr(13) & Chr(10) & _
            "The data should be arranged with a specific format.  The data must be ordered as such:" & Chr(13) & Chr(10) & _
            """Symbol,Quantity,Price,Action,TradeDate,Principal,Commission,SECFees,NetAmount,AccountType,Memo""" _
            & Chr(13) & Chr(10) & Chr(13) & Chr(10) & "If you feel confident the inputted file is reconciled and ready to processed, LClick on the" & Chr(13) & Chr(10) & _
            """Process Completely and Display Report"" button.  If you're not familiar with the file or need to remove" & Chr(13) & Chr(10) & _
            "open or failed issues then LClick the ""Run in Stages and View Steps"" button." & Chr(13) & Chr(10) & Chr(13) & Chr(10) & _
            "In either case Input Boxes will appear requesting inclusive start and stop date information for the final report." & Chr(13) & Chr(10) & _
            "Typical entries might be ""01/01/2003"" for a start date, and ""01/01/02004"" for an end date.  If the input file" & Chr(13) & Chr(10) & _
            "has not been examined you might be better off entering an very early start date and a future end date.  This allows" & Chr(13) & Chr(10) & _
            "observing the entire input file with Run in Stages and View Steps." & Chr(13) & Chr(10) & _
            "The Process Completely and Display Report button will produce a report and place it in the Report Text Box." & Chr(13) & Chr(10) & _
            "Several lines of report statistics proceed the processed input file data." & Chr(13) & Chr(10) & _
            "The line: ""File processing error information: Open Remaining Shares=  0, Accumulated Remnants Shares=  0"" " & Chr(13) & Chr(10) & _
            "should have 0 values.  The remnant issues can be observed by using the scroll bars to look through the " & Chr(13) & Chr(10) & _
            "groups of issues in the report file." & Chr(13) & Chr(10) & Chr(13) & Chr(10) & _
            """After the Run in Stages and View Steps"" process is explained a brief description of the Report data will be given." & Chr(13) & Chr(10) & _
            "The processing and calculation is broken down with ""Run in Stages and View Steps"" and several buttons appear."
    txtTextBox.Text = help_file
End Sub

Private Sub cmdHelpClose_Click()
    cmdHelpClose.Visible = False
    cmdHelp.Visible = True
    lblTextBox.Caption = ""
    txtTextBox.Text = ""  'You could try to reload Report string
End Sub

Private Sub cmdInputFilespec_Click()
```

```
    lblCopyright.Visible = False
    dlgInFiles.Filter = "All Files (*.*)|*.*|Comma Delimited Files (*.csv)|*.txt"
  ' Specify default filter.
    dlgInFiles.FilterIndex = 2

  ' Display the Open dialog box.
    dlgInFiles.ShowOpen
    InputFileSpec = dlgInFiles.FileName
    cmdInputFilespec.Visible = False

    Call BuildArray(Array_in(), val_row, InputFileSpec)

End Sub

Private Sub cmdOutputFileSpec_Click()
    dlgOutFiles.Filter = "Save Files "
  ' Specify default filter.
    dlgOutFiles.FilterIndex = 2

  ' Display the Save dialog box.
    dlgOutFiles.ShowSave
    OutputFileSpec = dlgOutFiles.FileName

Open dlgOutFiles.FileName For Output As #2

   Dim i As Integer, j As Integer, tmp_str As String, tmp_str1 As String, CRLF1 As String

    CRLF1 = Chr(13) & Chr(10)
    tmp_str = array_str

    Do While (Chr(13) = Left(tmp_str, 1) Or Chr(10) = Left(tmp_str, 1))
        tmp_str = Mid(tmp_str, 2)
    Loop

    Do
        j = InStr(tmp_str, CRLF1) + 1  'Find next CRLF

        tmp_str1 = Left(tmp_str, j)     'Get the line you want
        Print #2, Mid(tmp_str1, 1, InStr(tmp_str1, CRLF1))           'Place in the file
        tmp_str = Mid(tmp_str, j + 1) 'Get all the string past this CRLF for later

    Loop Until (10 > Len(tmp_str))

 Close #2

End Sub
```