```
Form_buyForm - 1

Option Compare Database

Public Sub VerticalAlignCenter(ByRef ctl As Control)
    Dim MinimumMargin As Integer
    Dim BorderWidth As Integer
    Dim TwipsPerPoint

    TwipsPerPoint = 20
    If Not ((TypeOf ctl Is TextBox) Or (TypeOf ctl Is Label)) Then Exit Sub

    'Figure out how many lines it is
    Dim LenOfText, WidOfBox, NumberOfLines, HtOfText
    If TypeOf ctl Is TextBox Then
        LenOfText = ctl.Text
    Else:
        LenOfText = ctl.Caption
    End If

    WidOfBox = ctl.Width
    LenOfText = (Len(LenOfText) * TwipsPerPoint * ctl.FontSize) / 2
    NumberOfLines = Int(LenOfText / WidOfBox) + 1
    HtOfText = NumberOfLines * TwipsPerPoint * ctl.FontSize


    MinimumMargin = 1 * TwipsPerPoint
    BorderWidth = (ctl.BorderWidth * TwipsPerPoint) / 2

    ctl.TopMargin = ((ctl.Height - HtOfText) / 2) - MinimumMargin - BorderWidth
End Sub

Private Sub backButton_Click()
    DoCmd.Close acForm, "buyForm", acSaveYes
    DoCmd.OpenForm "mainForm"
End Sub

Private Sub Form_Load()
    Me.itemDescription.Visible = False
    Me.sellerInfo.Visible = False
    Me.pictureBox.Visible = True

    DoCmd.OpenForm "loginForm", , , , , acHidden
    Dim SQL As String

    SQL = "SELECT * from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    Me.currentBalanceLabel.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("currentBal
ance").value, 2)
    Forms![loginForm].Visible = False
    'CurrentDb.OpenRecordset(SQL, dbOpenSnapshot, dbReadOnly)
    ' Used to manipulate data faster, as it is only a ReadOnly and 'snapshot'
    ' Concatenate (bring multiple pieces together)
    ' Populate Username
    Me.loggedInUser.Caption = CurrentDb.OpenRecordset(SQL).Fields("firstName").value + " " + Curren
tDb.OpenRecordset(SQL, dbOpenSnapshot, dbReadOnly).Fields("lastName").value
    Me.userID.Caption = CurrentDb.OpenRecordset(SQL).Fields("userID").value
    VerticalAlignCenter Me.itemDescription

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True
End Sub
Private Sub sellItem(itemID As Integer, soldPrice As Currency)
    Dim baseSQL As String
    Dim bidSQL As String
    Dim sellSQL As String

    ' Do first test to check database for any bids on item
    bidSQL = "Select userTable.userID, userTable.currentBalance, bidTable.bidTime, bidTable.price "
 & _
             "FROM userTable INNER JOIN bidTable ON userTable.userID = bidTable.userID " & _
             "WHERE bidTable.itemID = " & itemID & " " & _
             "ORDER BY bidTable.bidTime DESC;"

    baseSQL = "SELECT itemEntity.*, userTable.* " & _
```

```
                "FROM userTable INNER JOIN itemEntity ON userTable.userID = itemEntity.sellerID "
 & _
                "WHERE itemEntity.itemID = " & itemID

    ' Set latest Bid
    If CurrentDb.OpenRecordset(bidSQL).EOF Then
        ' No bid has been made, sell back to seller
        ' Find SellerID
        sellSQL = "INSERT INTO soldItems (userID, itemID, soldPrice) " & _
                "VALUES (" & CurrentDb.OpenRecordset(baseSQL).Fields("sellerID").value & ", " & i
temID & ", " & soldPrice & ");"

        CurrentDb.Execute sellSQL
        ' Note no monetary difference has been made from this transaction
    Else
        ' Bid has been made; sell to highest bidder
        ' Find buyerID
        sellSQL = "INSERT INTO soldItems (userID, itemID, soldPrice) " & _
                "VALUES (" & CurrentDb.OpenRecordset(bidSQL).Fields("userID") & ", " & itemID & "
, " & soldPrice & ");"
        CurrentDb.Execute sellSQL
        ' Add money to seller, subtract from buyer
        Dim owing As Currency
        owing = CurrentDb.OpenRecordset(bidSQL).Fields("currentBalance") - soldPrice

        sellSQL = "UPDATE userTable " & _
                "SET currentBalance=" + CStr(owing) + " " & _
                "WHERE userID=" + CStr(CurrentDb.OpenRecordset(bidSQL).Fields("userID")) + ";"

        CurrentDb.Execute sellSQL

        owing = CurrentDb.OpenRecordset(baseSQL).Fields("currentBalance") + soldPrice
        sellSQL = "UPDATE userTable " & _
                "SET currentBalance=" + CStr(owing) + " " & _
                "WHERE userID=" + CStr(CurrentDb.OpenRecordset(baseSQL).Fields("sellerID")) + ";"
        CurrentDb.Execute sellSQL

        sellSQL = "SELECT * from itemEntity WHERE itemID=" + CStr(itemID) + ";"

        If Me.userID.Caption = CStr(CurrentDb.OpenRecordset(bidSQL).Fields("userID")) Then
            MsgBox "Congratulations! You have won " + CurrentDb.OpenRecordset(sellSQL).Fields("item
Name") + "."
        ElseIf Me.userID.Caption = CStr(CurrentDb.OpenRecordset(baseSQL).Fields("userID")) Then
            MsgBox "Congratulations! " + CurrentDb.OpenRecordset(sellSQL).Fields("itemName") + " ha
s been successfully sold."
        Else
            ' Person logged in didn't win anything, do nothing
        End If
    End If

    Me.itemListBox.Requery

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True

    updateItems
End Sub
Private Sub Form_Timer()
    ' Continuously update values, such as bid price and the timing
    ' Check if currently anything selected

    If IsNull(Me.itemListBox.Column(0)) Then
        ' Do nothing
        Me.bidLength.Caption = "-"
    Else
        Dim SQL As String
        Dim bidSQL As String

        SQL = "SELECT itemEntity.*, userTable.* " & _
          "FROM userTable INNER JOIN itemEntity ON userTable.userID = itemEntity.sellerID " & _
          "WHERE itemEntity.itemID = " & Me.itemListBox.Column(0)

        bidSQL = "Select userTable.userName,bidTable.bidTime, bidTable.price " & _
             "FROM userTable INNER JOIN bidTable ON userTable.userID = bidTable.userID " & _
             "WHERE bidTable.itemID = " & Me.itemListBox.Column(0) & " " & _
```

```
             "ORDER BY bidTable.bidTime DESC;"
        ' Update Latest bid BEFORE updating timing interval
        If Me.currentBid.Caption = "currentPrice" Then
            ' Set item description
            Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").val
ue
            ' Set image
            Me.sellerInfo.Caption = CurrentDb.OpenRecordset(SQL).Fields("userName").value
            Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
            ' Set latest Bid
            If CurrentDb.OpenRecordset(bidSQL).EOF Then
                ' Set to initial Bid as none have been made yet
                Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("initBid
").value, 2)
            Else
                ' Set to latest Bid
                Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(bidSQL).Fields("pric
e").value, 2)
            End If
        Else
            ' Set latest Bid
            If CurrentDb.OpenRecordset(bidSQL).EOF Then
                ' Set to initial Bid as none have been made yet
                Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("initBid
").value, 2)
            Else
                ' Set to latest Bid
                Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(bidSQL).Fields("pric
e").value, 2)
            End If
                ' Do nothing
        End If


        If Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value The
n
            ' Already called db
            Dim shortDate As Date
            Dim shortTime As Date

            shortDate = Format(Me.hiddenBidLength.Caption, "dd/mm/yyyy")
            shortTime = Format(Me.hiddenBidLength.Caption, "hh:mm:ss")

            Dim dblNumDays  As Integer
            Dim dblNumHours As Integer
            Dim dblNumMins  As Integer
            Dim dblNumSeconds As Integer

            dblNumDays = DateDiff("d", Format(Now(), "dd/mm/yyyy"), shortDate)
            If dblNumDays > 0 Then
                ' Positive Number
                shortTime = Format("23:59:59", "hh:mm:ss")
            Else
                ' Do nothing
            End If
            dblNumHours = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) / 3600
            dblNumMins = (((DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime)) / 60)) Mod 60
            dblNumSeconds = (DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) + 30) Mod 60

            Me.bidLength.Caption = dblNumDays & ":" & dblNumHours & ":" & dblNumMins & ":" & dblNum
Seconds

            If dblNumSeconds < 0 Then
                ' Item should be sold, sell to highest bidder (and/or back to seller)
                sellItem Me.itemListBox.Column(0), FormatCurrency(Me.currentBid.Caption, 2)
            Else
                ' Do nothing
            End If
        Else
            ' Haven't called Database
            ' Set hidden label to be respective database value
            Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value
        End If
    End If

End Sub
```

```
Private Sub itemDescription_Click()
    Me.itemListBox.SetFocus
    Me.itemDescription.Visible = False
    Me.sellerInfo.Visible = False
    Me.pictureBox.Visible = True
End Sub
Private Sub updateItems()
On Error GoTo errUpdateItems
    ' Now that row value has been found (representing itemID in itemEntityTable),
    ' Call SQL query to populate other fields.
    Dim SQL As String
    Dim bidSQL As String

    SQL = "SELECT itemEntity.*, userTable.* " & _
          "FROM userTable INNER JOIN itemEntity ON userTable.userID = itemEntity.sellerID " & _
          "WHERE itemEntity.itemID = " & Me.itemListBox.Column(0)

    bidSQL = "Select userTable.userName,bidTable.bidTime, bidTable.price " & _
             "FROM userTable INNER JOIN bidTable ON userTable.userID = bidTable.userID " & _
             "WHERE bidTable.itemID = " & Me.itemListBox.Column(0) & " " & _
             "ORDER BY bidTable.bidTime DESC;"
    ' NEED:
    ' Item Description
    ' Picture Location
    ' initBid
    ' endBidTime

    ' Set item description
    Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").value
    ' Set button text
    Me.sellerInfo.Caption = CurrentDb.OpenRecordset(SQL).Fields("userName").value
    ' Set image
    Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
    ' Set latest Bid
    If CurrentDb.OpenRecordset(bidSQL).EOF Then
        ' Set to initial Bid as none have been made yet
        Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("initBid").value
, 2)
    Else
        ' Set to latest Bid
        Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(bidSQL).Fields("price").valu
e, 2)
    End If
    ' Set bid time difference
    Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value
    If Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value Then
        ' Already called db
        Dim shortDate As Date
        Dim shortTime As Date

        shortDate = Format(Me.hiddenBidLength.Caption, "dd/mm/yyyy")
        shortTime = Format(Me.hiddenBidLength.Caption, "hh:mm:ss")

        Dim dblNumDays  As Integer
        Dim dblNumHours As Integer
        Dim dblNumMins  As Integer
        Dim dblNumSeconds As Integer

        dblNumDays = DateDiff("d", Format(Now(), "dd/mm/yyyy"), shortDate)
        If dblNumDays > 0 Then
            ' Positive Number
            shortTime = Format("23:59:59", "hh:mm:ss")
        Else
            ' Do nothing
        End If
        dblNumHours = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) / 3600
        dblNumMins = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) / 60 Mod 60
        dblNumSeconds = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) Mod 60

        Me.bidLength.Caption = dblNumDays & ":" & dblNumHours & ":" & dblNumMins & ":" & dblNumSeco
nds
    Else
        ' Haven't called Database
        ' Set hidden label to be respective database value
        Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value
```

```
    End If

errUpdateItems:
    Exit Sub
End Sub
Private Sub itemListBox_AfterUpdate()
    updateItems
End Sub

Private Sub pictureBox_Click()
    Me.itemDescription.Visible = True
    Me.sellerInfo.Visible = True
    Me.pictureBox.Visible = False
End Sub

Private Sub placeBid_Click()
    ' Place Bid
    If IsNull(Me.bidInput.value) Then
        ' Textbox Empty
        MsgBox "Please enter a value to bid"
    Else
        Dim inputtedValue As Double
        Dim latestBid As Double
        Dim currentBalance As Double

        inputtedValue = CStr(Format(Me.bidInput.value, "General Number"))
        latestBid = CStr(Format(Me.currentBid.Caption, "General Number"))
        currentBalance = CStr(Format(Me.currentBalanceLabel.Caption, "General Number"))

        If inputtedValue < latestBid Then
            ' Lower than highest bid
            MsgBox "Please Enter a higher bid"
        Else
            If inputtedValue < currentBalance Then
                ' Value within price range of account (able to be done)
                Dim SQL As String
                Me.bidInput.SetFocus
                SQL = "INSERT INTO bidTable (userID, itemID, price, bidTime) " & _
                  "VALUES (" & Me.userID.Caption & ", " & Me.itemListBox.Column(0) & ", " & CStr(Fo
rmat(Me.bidInput.Text, "General Number")) & ", '" & Now() & "');"

                CurrentDb.Execute SQL

                Me.bidInput.value = ""
                Me.placeBid.SetFocus

                MsgBox "Bid successfully placed"
            Else
                MsgBox "Too much money, can't afford"
            End If
        End If
    End If
End Sub

Private Sub searchBox_Change()
    Dim vSearchString As String

    vSearchString = searchBox.Text

    searchHiddenInput.value = vSearchString

    Me.itemListBox.Requery

    If Len(Me.searchHiddenInput) <> 0 And InStr(Len(searchHiddenInput), searchHiddenInput, " ", vbT
extCompare) Then
        Exit Sub
    End If

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True

    updateItems

    Me.searchBox.SetFocus
```

```
    If Not IsNull(Len(Me.searchBox)) Then
        Me.searchBox.SelStart = Len(Me.searchBox)
    End If



End Sub

Private Sub sellerInfo_Click()
    ' Find seller ID from seller UserName
    SQL = "SELECT itemEntity.*, userTable.* " & _
          "FROM userTable INNER JOIN itemEntity ON userTable.userID = itemEntity.sellerID " & _
          "WHERE itemEntity.itemID = " & Me.itemListBox.Column(0)

    DoCmd.OpenForm "userProfileForm", , , "userID = " & CInt(CurrentDb.OpenRecordset(SQL).Fields("u
serID").value)

End Sub
```

```
Form_loginForm - 1

Option Compare Database

Option Explicit
Option Base 0

' A VB6/VBA procedure for the MD5 message-digest algorithm
' as described in RFC 1321 by R. Rivest, April 1992

Private Const MD5_BLK_LEN As Long = 64
' Constants for MD5Transform routine
Private Const S11 As Long = 7
Private Const S12 As Long = 12
Private Const S13 As Long = 17
Private Const S14 As Long = 22
Private Const S21 As Long = 5
Private Const S22 As Long = 9
Private Const S23 As Long = 14
Private Const S24 As Long = 20
Private Const S31 As Long = 4
Private Const S32 As Long = 11
Private Const S33 As Long = 16
Private Const S34 As Long = 23
Private Const S41 As Long = 6
Private Const S42 As Long = 10
Private Const S43 As Long = 15
Private Const S44 As Long = 21
' Constants for unsigned word addition
Private Const OFFSET_4 = 4294967296#
Private Const MAXINT_4 = 2147483647

' TEST FUNCTIONS...
' MD5 test suite:
' MD5 ("") = d41d8cd98f00b204e9800998ecf8427e
' MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661
' MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72
' MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0
' MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b
' MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") =
' d174ab98d277d9f5a5611c2c9f419d9f
' MD5 ("12345678901234567890123456789012345678901234567890123456789012345678
' 78901234567890") = 57edf4a22be3c955ac49da2e2107b67a

' MD5 (1 million x 'a') = 7707d6ae4e027c70eea2a935c2296f21

Public Function Test_md5_abc()
    Debug.Print MD5_string("abc")
End Function

Public Function md5_test_suite()
    Debug.Print MD5_string("")
    Debug.Print MD5_string("a")
    Debug.Print MD5_string("abc")
    Debug.Print MD5_string("message digest")
    Debug.Print MD5_string("abcdefghijklmnopqrstuvwxyz")
    Debug.Print MD5_string("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789")
    Debug.Print MD5_string("12345678901234567890123456789012345678901234567890123456789012345678901
234567890")
End Function

Public Function test_md5_empty()
    Debug.Print MD5_string("")
End Function

Public Function test_md5_around64()
    Dim strMessage As String
    strMessage = "12345678901234567890123456789012345678901234567890123456789012345678901234567890"
    Debug.Print MD5_string(strMessage)
    Debug.Print MD5_string(Left(strMessage, 65))
    Debug.Print MD5_string(Left(strMessage, 64))
    Debug.Print MD5_string(Left(strMessage, 63))
    Debug.Print MD5_string(Left(strMessage, 62))
    Debug.Print MD5_string(Left(strMessage, 57))
    Debug.Print MD5_string(Left(strMessage, 56))
    Debug.Print MD5_string(Left(strMessage, 55))
End Function
```

```
Form_loginForm - 2

Public Function test_md5_million_a()
' This may take some time...
    Dim abMessage() As Byte
    Dim mLen As Long
    Dim i As Long
    mLen = 1000000
    ReDim abMessage(mLen - 1)
    For i = 0 To mLen - 1
        abMessage(i) = &H61     ' 0x61 = 'a'
    Next
    Debug.Print MD5_bytes(abMessage, mLen)

End Function

' MAIN EXPORTED MD5 FUNCTIONS...

Public Function MD5_string(strMessage As String) As String
' Returns 32-char hex string representation of message digest
' Input as a string (max length 2^29-1 bytes)
    Dim abMessage() As Byte
    Dim mLen As Long
    ' Cope with the empty string
    If Len(strMessage) > 0 Then
        abMessage = StrConv(strMessage, vbFromUnicode)
        ' Compute length of message in bytes
        mLen = UBound(abMessage) - LBound(abMessage) + 1
    End If
    MD5_string = MD5_bytes(abMessage, mLen)
End Function

Public Function MD5_bytes(abMessage() As Byte, mLen As Long) As String
' Returns 32-char hex string representation of message digest
' Input as an array of bytes of length mLen bytes

    Dim nBlks As Long
    Dim nBits As Long
    Dim block(MD5_BLK_LEN - 1) As Byte
    Dim state(3) As Long
    Dim wb(3) As Byte
    Dim sHex As String
    Dim index As Long
    Dim partLen As Long
    Dim i As Long
    Dim j As Long

    ' Catch length too big for VB arithmetic (268 million!)
    If mLen >= &HFFFFFFF Then Error 6     ' overflow

    ' Initialise
    ' Number of complete 512-bit/64-byte blocks to process
    nBlks = mLen \ MD5_BLK_LEN

    ' Load magic initialization constants
    state(0) = &H67452301
    state(1) = &HEFCDAB89
    state(2) = &H98BADCFE
    state(3) = &H10325476

    ' Main loop for each complete input block of 64 bytes
    index = 0
    For i = 0 To nBlks - 1
        Call md5_transform(state, abMessage, index)
        index = index + MD5_BLK_LEN
    Next

    ' Construct final block(s) with padding
    partLen = mLen Mod MD5_BLK_LEN
    index = nBlks * MD5_BLK_LEN
    For i = 0 To partLen - 1
        block(i) = abMessage(index + i)
    Next
    block(partLen) = &H80
    ' Make sure padding (and bit-length) set to zero
    For i = partLen + 1 To MD5_BLK_LEN - 1
        block(i) = 0
    Next
```

```
    ' Two cases: partLen is < or >= 56
    If partLen >= MD5_BLK_LEN - 8 Then
        ' Need two blocks
        Call md5_transform(state, block, 0)
        For i = 0 To MD5_BLK_LEN - 1
            block(i) = 0
        Next
    End If
    ' Append number of bits in little-endian order
    nBits = mLen * 8
    block(MD5_BLK_LEN - 8) = nBits And &HFF
    block(MD5_BLK_LEN - 7) = nBits \ &H100 And &HFF
    block(MD5_BLK_LEN - 6) = nBits \ &H10000 And &HFF
    block(MD5_BLK_LEN - 5) = nBits \ &H1000000 And &HFF
    ' (NB we don't try to cope with number greater than 2^31)

    ' Final padded block with bit length
    Call md5_transform(state, block, 0)

    ' Decode 4 x 32-bit words into 16 bytes with LSB first each time
    ' and return result as a hex string
    MD5_bytes = ""
    For i = 0 To 3
        Call uwSplit(state(i), wb(3), wb(2), wb(1), wb(0))
        For j = 0 To 3
            If wb(j) < 16 Then
                sHex = "0" & Hex(wb(j))
            Else
                sHex = Hex(wb(j))
            End If
            MD5_bytes = MD5_bytes & sHex
        Next
    Next

End Function

' INTERNAL FUNCTIONS...

Private Sub md5_transform(state() As Long, buf() As Byte, ByVal index As Long)
' Updates 4 x 32-bit values in state
' Input: the next 64 bytes in buf starting at offset index
' Assumes at least 64 bytes are present after offset index
    Dim a As Long
    Dim b As Long
    Dim c As Long
    Dim d As Long
    Dim j As Integer
    Dim x(15) As Long

    a = state(0)
    b = state(1)
    c = state(2)
    d = state(3)

    ' Decode the next 64 bytes into 16 words with LSB first
    For j = 0 To 15
        x(j) = uwJoin(buf(index + 3), buf(index + 2), buf(index + 1), buf(index))
        index = index + 4
    Next

    ' Round 1
    a = FF(a, b, c, d, x(0), S11, &HD76AA478)   ' 1
    d = FF(d, a, b, c, x(1), S12, &HE8C7B756)   ' 2
    c = FF(c, d, a, b, x(2), S13, &H242070DB)   ' 3
    b = FF(b, c, d, a, x(3), S14, &HC1BDCEEE)   ' 4
    a = FF(a, b, c, d, x(4), S11, &HF57C0FAF)   ' 5
    d = FF(d, a, b, c, x(5), S12, &H4787C62A)   ' 6
    c = FF(c, d, a, b, x(6), S13, &HA8304613)   ' 7
    b = FF(b, c, d, a, x(7), S14, &HFD469501)   ' 8
    a = FF(a, b, c, d, x(8), S11, &H698098D8)   ' 9
    d = FF(d, a, b, c, x(9), S12, &H8B44F7AF)   ' 10
    c = FF(c, d, a, b, x(10), S13, &HFFFF5BB1)  ' 11
    b = FF(b, c, d, a, x(11), S14, &H895CD7BE)  ' 12
    a = FF(a, b, c, d, x(12), S11, &H6B901122)  ' 13
    d = FF(d, a, b, c, x(13), S12, &HFD987193)  ' 14
    c = FF(c, d, a, b, x(14), S13, &HA679438E)  ' 15
```

```
Form_loginForm - 4

    b = FF(b, c, d, a, x(15), S14, &H49B40821)  ' 16

    ' Round 2
    a = GG(a, b, c, d, x(1), S21, &HF61E2562)   ' 17
    d = GG(d, a, b, c, x(6), S22, &HC040B340)   ' 18
    c = GG(c, d, a, b, x(11), S23, &H265E5A51)  ' 19
    b = GG(b, c, d, a, x(0), S24, &HE9B6C7AA)   ' 20
    a = GG(a, b, c, d, x(5), S21, &HD62F105D)   ' 21
    d = GG(d, a, b, c, x(10), S22, &H2441453)   ' 22
    c = GG(c, d, a, b, x(15), S23, &HD8A1E681)  ' 23
    b = GG(b, c, d, a, x(4), S24, &HE7D3FBC8)   ' 24
    a = GG(a, b, c, d, x(9), S21, &H21E1CDE6)   ' 25
    d = GG(d, a, b, c, x(14), S22, &HC33707D6)  ' 26
    c = GG(c, d, a, b, x(3), S23, &HF4D50D87)   ' 27
    b = GG(b, c, d, a, x(8), S24, &H455A14ED)   ' 28
    a = GG(a, b, c, d, x(13), S21, &HA9E3E905)  ' 29
    d = GG(d, a, b, c, x(2), S22, &HFCEFA3F8)   ' 30
    c = GG(c, d, a, b, x(7), S23, &H676F02D9)   ' 31
    b = GG(b, c, d, a, x(12), S24, &H8D2A4C8A)  ' 32

    ' Round 3
    a = HH(a, b, c, d, x(5), S31, &HFFFA3942)   ' 33
    d = HH(d, a, b, c, x(8), S32, &H8771F681)   ' 34
    c = HH(c, d, a, b, x(11), S33, &H6D9D6122)  ' 35
    b = HH(b, c, d, a, x(14), S34, &HFDE5380C)  ' 36
    a = HH(a, b, c, d, x(1), S31, &HA4BEEA44)   ' 37
    d = HH(d, a, b, c, x(4), S32, &H4BDECFA9)   ' 38
    c = HH(c, d, a, b, x(7), S33, &HF6BB4B60)   ' 39
    b = HH(b, c, d, a, x(10), S34, &HBEBFBC70)  ' 40
    a = HH(a, b, c, d, x(13), S31, &H289B7EC6)  ' 41
    d = HH(d, a, b, c, x(0), S32, &HEAA127FA)   ' 42
    c = HH(c, d, a, b, x(3), S33, &HD4EF3085)   ' 43
    b = HH(b, c, d, a, x(6), S34, &H4881D05)    ' 44
    a = HH(a, b, c, d, x(9), S31, &HD9D4D039)   ' 45
    d = HH(d, a, b, c, x(12), S32, &HE6DB99E5)  ' 46
    c = HH(c, d, a, b, x(15), S33, &H1FA27CF8)  ' 47
    b = HH(b, c, d, a, x(2), S34, &HC4AC5665)   ' 48

    ' Round 4
    a = II(a, b, c, d, x(0), S41, &HF4292244)   ' 49
    d = II(d, a, b, c, x(7), S42, &H432AFF97)   ' 50
    c = II(c, d, a, b, x(14), S43, &HAB9423A7)  ' 51
    b = II(b, c, d, a, x(5), S44, &HFC93A039)   ' 52
    a = II(a, b, c, d, x(12), S41, &H655B59C3)  ' 53
    d = II(d, a, b, c, x(3), S42, &H8F0CCC92)   ' 54
    c = II(c, d, a, b, x(10), S43, &HFFEFF47D)  ' 55
    b = II(b, c, d, a, x(1), S44, &H85845DD1)   ' 56
    a = II(a, b, c, d, x(8), S41, &H6FA87E4F)   ' 57
    d = II(d, a, b, c, x(15), S42, &HFE2CE6E0)  ' 58
    c = II(c, d, a, b, x(6), S43, &HA3014314)   ' 59
    b = II(b, c, d, a, x(13), S44, &H4E0811A1)  ' 60
    a = II(a, b, c, d, x(4), S41, &HF7537E82)   ' 61
    d = II(d, a, b, c, x(11), S42, &HBD3AF235)  ' 62
    c = II(c, d, a, b, x(2), S43, &H2AD7D2BB)   ' 63
    b = II(b, c, d, a, x(9), S44, &HEB86D391)   ' 64

    state(0) = uwAdd(state(0), a)
    state(1) = uwAdd(state(1), b)
    state(2) = uwAdd(state(2), c)
    state(3) = uwAdd(state(3), d)

End Sub

' FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4

Private Function AddRotAdd(f As Long, a As Long, b As Long, x As Long, s As Integer, ac As Long) As
 Long
' Common routine for FF, GG, HH and II
' #define AddRotAdd(f, a, b, c, d, x, s, ac) { \
'  (a) += f + (x) + (UINT4)(ac); \
'  (a) = ROTATE_LEFT ((a), (s)); \
'  (a) += (b); \
'  }
    Dim temp As Long
    temp = uwAdd(a, f)
    temp = uwAdd(temp, x)
```

```
    temp = uwAdd(temp, ac)
    temp = uwRol(temp, s)
    AddRotAdd = uwAdd(temp, b)
End Function

Private Function FF(a As Long, b As Long, c As Long, d As Long, x As Long, s As Integer, ac As Long
) As Long
' Returns new value of a
' #define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
' #define FF(a, b, c, d, x, s, ac) { \
'  (a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \
'  (a) = ROTATE_LEFT ((a), (s)); \
'  (a) += (b); \
'  }
    Dim t As Long
    Dim t2 As Long
    ' F ((b), (c), (d)) = (((b) & (c)) | ((~b) & (d)))
    t = b And c
    t2 = (Not b) And d
    t = t Or t2
    FF = AddRotAdd(t, a, b, x, s, ac)
End Function

Private Function GG(a As Long, b As Long, c As Long, d As Long, x As Long, s As Integer, ac As Long
) As Long
' #define G(b, c, d) (((b) & (d)) | ((c) & (~d)))
    Dim t As Long
    Dim t2 As Long
    t = b And d
    t2 = c And (Not d)
    t = t Or t2
    GG = AddRotAdd(t, a, b, x, s, ac)
End Function

Private Function HH(a As Long, b As Long, c As Long, d As Long, x As Long, s As Integer, ac As Long
) As Long
' #define H(b, c, d) ((b) ^ (c) ^ (d))
    Dim t As Long
    t = b Xor c Xor d
    HH = AddRotAdd(t, a, b, x, s, ac)
End Function

Private Function II(a As Long, b As Long, c As Long, d As Long, x As Long, s As Integer, ac As Long
) As Long
' #define I(b, c, d) ((c) ^ ((b) | (~d)))
    Dim t As Long
    t = b Or (Not d)
    t = c Xor t
    II = AddRotAdd(t, a, b, x, s, ac)
End Function

' Unsigned 32-bit word functions suitable for VB/VBA

Private Function uwRol(w As Long, s As Integer) As Long
' Return 32-bit word w rotated left by s bits
' avoiding problem with VB sign bit
    Dim i As Integer
    Dim t As Long

    uwRol = w
    For i = 1 To s
        t = uwRol And &H3FFFFFFF
        t = t * 2
        If (uwRol And &H40000000) <> 0 Then
            t = t Or &H80000000
        End If
        If (uwRol And &H80000000) <> 0 Then
            t = t Or &H1
        End If
        uwRol = t
    Next
End Function

Private Function uwJoin(a As Byte, b As Byte, c As Byte, d As Byte) As Long
' Join 4 x 8-bit bytes into one 32-bit word a.b.c.d
    uwJoin = ((a And &H7F) * &H1000000) Or (b * &H10000) Or (CLng(c) * &H100) Or d
```

```
    If a And &H80 Then
        uwJoin = uwJoin Or &H80000000
    End If
End Function

Private Sub uwSplit(ByVal w As Long, a As Byte, b As Byte, c As Byte, d As Byte)
' Split 32-bit word w into 4 x 8-bit bytes
    a = CByte(((w And &HFF000000) \ &H1000000) And &HFF)
    b = CByte(((w And &HFF0000) \ &H10000) And &HFF)
    c = CByte(((w And &HFF00) \ &H100) And &HFF)
    d = CByte((w And &HFF) And &HFF)
End Sub

Public Function uwAdd(wordA As Long, wordB As Long) As Long
' Adds words A and B avoiding overflow
    Dim myUnsigned As Double

    myUnsigned = LongToUnsigned(wordA) + LongToUnsigned(wordB)
    ' Cope with overflow
    '[2010-10-20] Changed from ">" to ">=". Thanks Loek.
    If myUnsigned >= OFFSET_4 Then
        myUnsigned = myUnsigned - OFFSET_4
    End If
    uwAdd = UnsignedToLong(myUnsigned)

End Function

'***************************************************
' These two functions from Microsoft Article Q189323
' "HOWTO: convert between Signed and Unsigned Numbers"

Private Function UnsignedToLong(value As Double) As Long
    If value < 0 Or value >= OFFSET_4 Then Error 6 ' Overflow
    If value <= MAXINT_4 Then
        UnsignedToLong = value
    Else
        UnsignedToLong = value - OFFSET_4
    End If
End Function

Private Function LongToUnsigned(value As Long) As Double
    If value < 0 Then
        LongToUnsigned = value + OFFSET_4
    Else
        LongToUnsigned = value
    End If
End Function

' End of Microsoft-article functions
'***************************************************


Private Sub verifyLogin()
    If IsNull(Me.usernameTextBox) Then
        ' No username Entered
        MsgBox "Please Enter Value for Username", vbInformation, "Username required"
        Me.usernameTextBox.SetFocus
    ElseIf IsNull(Me.passwordTextBox) Then
        ' No password Entered
        MsgBox "Please Enter Value for Password", vbInformation, "Password required"
        Me.passwordTextBox.SetFocus
    Else
        ' Username and Password entered in relevant fields, process and authenticate
        If (IsNull(DLookup("userName", "userTable", "[userName] ='" & Me.usernameTextBox.value & "'
 And [loggedIn] = 0"))) Or _
        (IsNull(DLookup("password", "userTable", "[password] ='" & MD5_string(Me.passwordTextBox.va
lue) & "'"))) Then
            MsgBox "Incorrect Login, try again"
        Else
            MsgBox "Login Successful, Please continue"

            ' Execute SQL updating who is currently logged into database
            Dim SQL As String

            SQL = "UPDATE userTable " & _
                  "SET userTable.loggedIn = 1 " & _
```

```
                  "WHERE userTable.userName ='" & Me.usernameTextBox.value & "'"

            CurrentDb.Execute SQL

            Forms![loginForm].Visible = False
            DoCmd.OpenForm "mainForm"
        End If
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Used to test if query will run that will run before exiting
    ' MsgBox "Sure you want to quit?", vbCritical, "Exit Confirmation"

    ' Run sql which will reset all logged in users
    Dim SQL As String

    SQL = "UPDATE userTable " & _
          "SET userTable.loggedIn = 0"

    CurrentDb.Execute SQL

End Sub

Private Sub loginButton_Click()
    verifyLogin
End Sub

Private Sub registerButton_Click()
    DoCmd.OpenForm "registerForm"
End Sub
```

Form_mainForm - 1

```
Option Compare Database

Private Sub buyButton_Click()
    DoCmd.Close acForm, "mainForm", acSaveYes
    DoCmd.OpenForm "buyForm"
End Sub

Private Sub Form_Load()
    DoCmd.OpenForm "loginForm", , , , , acHidden
    Dim SQL As String

    SQL = "SELECT * from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    Me.loggedInUser.Caption = CurrentDb.OpenRecordset(SQL).Fields("firstName").value + " " + Curren
tDb.OpenRecordset(SQL).Fields("lastName").value
    Me.currentBalanceLabel.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("currentBal
ance").value, 2)
    Forms![loginForm].Visible = False
End Sub

Private Sub logoutButton_Click()
    Dim SQL As String
    DoCmd.OpenForm "loginForm"
    SQL = "UPDATE userTable " & _
          "SET userTable.loggedIn = 0 " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    CurrentDb.Execute SQL
    Forms![loginForm].Visible = False

    DoCmd.Close acForm, "mainForm", acSaveYes

    Forms![loginForm].usernameTextBox.value = ""
    Forms![loginForm].passwordTextBox.value = ""

    Forms![loginForm].usernameTextBox.SetFocus

    Forms![loginForm].Visible = True
End Sub

Private Sub profileButton_Click()
    DoCmd.Close acForm, "mainForm", acSaveYes
    DoCmd.OpenForm "profileForm"
End Sub

Private Sub purchasesButton_Click()
    DoCmd.Close acForm, "mainForm", acSaveYes
    DoCmd.OpenForm "purchasesForm"
End Sub

Private Sub salesButton_Click()
    DoCmd.Close acForm, "mainForm", acSaveYes
    DoCmd.OpenForm "soldForm"
End Sub

Private Sub sellButton_Click()
    DoCmd.Close acForm, "mainForm", acSaveYes
    DoCmd.OpenForm "sellForm"
End Sub
```

Form_profileForm - 1

```
Option Compare Database

Private Sub Form_Load()
    DoCmd.OpenForm "loginForm", , , , , acHidden

    Dim SQL As String
    SQL = "SELECT userID from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    DoCmd.GoToRecord acDataForm, Me.Name, acGoTo, CInt(CurrentDb.OpenRecordset(SQL).Fields("userID"
).value)

    Forms![loginForm].Visible = False
End Sub

Private Sub updateButton_Click()
    DoCmd.Close acForm, "profileForm", acSaveYes
    DoCmd.OpenForm "mainForm"
End Sub
```

```
Form_purchasesForm - 1

Option Compare Database

Public Sub VerticalAlignCenter(ByRef ctl As Control)
    Dim MinimumMargin As Integer
    Dim BorderWidth As Integer
    Dim TwipsPerPoint

    TwipsPerPoint = 20
    If Not ((TypeOf ctl Is TextBox) Or (TypeOf ctl Is Label)) Then Exit Sub

    'Figure out how many lines it is
    Dim LenOfText, WidOfBox, NumberOfLines, HtOfText
    If TypeOf ctl Is TextBox Then
        LenOfText = ctl.Text
    Else:
        LenOfText = ctl.Caption
    End If

    WidOfBox = ctl.Width
    LenOfText = (Len(LenOfText) * TwipsPerPoint * ctl.FontSize) / 2
    NumberOfLines = Int(LenOfText / WidOfBox) + 1
    HtOfText = NumberOfLines * TwipsPerPoint * ctl.FontSize


    MinimumMargin = 1 * TwipsPerPoint
    BorderWidth = (ctl.BorderWidth * TwipsPerPoint) / 2

    ctl.TopMargin = ((ctl.Height - HtOfText) / 2) - MinimumMargin - BorderWidth
End Sub

Private Sub backButton_Click()
    DoCmd.Close acForm, "purchasesForm", acSaveYes
    DoCmd.OpenForm "mainForm"
End Sub

Private Sub Form_Load()
    Me.itemDescription.Visible = False
    Me.pictureBox.Visible = True

    DoCmd.OpenForm "loginForm", , , , , acHidden
    Dim SQL As String

    SQL = "SELECT * from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    Me.currentBalanceLabel.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("currentBal
ance").value, 2)
    Forms![loginForm].Visible = False
    'CurrentDb.OpenRecordset(SQL, dbOpenSnapshot, dbReadOnly)
    ' Used to manipulate data faster, as it is only a ReadOnly and 'snapshot'
    ' Concatenate (bring multiple pieces together)
    ' Populate Username
    Me.loggedInUser.Caption = CurrentDb.OpenRecordset(SQL).Fields("firstName").value + " " + Curren
tDb.OpenRecordset(SQL, dbOpenSnapshot, dbReadOnly).Fields("lastName").value
    Me.userID.value = CurrentDb.OpenRecordset(SQL).Fields("userID").value
    VerticalAlignCenter Me.itemDescription

    Me.itemListBox.Requery

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True
End Sub

Private Sub Form_Timer()
    If IsNull(Me.itemListBox.Column(0)) Then
        ' Do nothing
        'Me.bidLength.Caption = "-"
    Else
        Dim SQL As String
        Dim bidSQL As String

        SQL = "SELECT soldItems.*, itemEntity.* " & _
          "FROM itemEntity INNER JOIN soldItems ON itemEntity.itemID = soldItems.itemID " & _
          "WHERE soldItems.itemID = " & Me.itemListBox.Column(0)
```

```
        ' Update Latest bid BEFORE updating timing interval
        If Me.currentBid.Caption = "currentPrice" Then
            ' Set item description
            Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").val
ue
            ' Set image
            Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
            ' Set price
            Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("soldPrice")
.value, 2)
        Else
            ' Do nothing
        End If
    End If
End Sub

Private Sub itemDescription_Click()
    Me.itemDescription.Visible = False
    Me.pictureBox.Visible = True
End Sub
Private Sub updateItems()
On Error GoTo errUpdateItems
    ' Now that row value has been found (representing itemID in itemEntityTable),
    ' Call SQL query to populate other fields.
    Dim SQL As String
    Dim bidSQL As String

    SQL = "SELECT soldItems.*, itemEntity.* " & _
          "FROM itemEntity INNER JOIN soldItems ON itemEntity.itemID = soldItems.itemID " & _
          "WHERE soldItems.itemID = " & Me.itemListBox.Column(0)
    ' NEED:
    ' Item Description
    ' Picture Location
    ' initBid
    ' endBidTime

    ' Set item description
    Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").value
    ' Set image
    Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
    ' Set latest Bid
    Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("soldPrice").value,
2)
errUpdateItems:
    Exit Sub
End Sub
Private Sub itemListBox_AfterUpdate()
    updateItems
End Sub

Private Sub pictureBox_Click()
    Me.itemDescription.Visible = True
    Me.pictureBox.Visible = False
End Sub

Private Sub placeBid_Click()
    ' Place Bid
    If IsNull(Me.bidInput.value) Then
        ' Textbox Empty
        MsgBox "Please enter a value to bid"
    Else
        Dim inputtedValue As Double
        Dim latestBid As Double
        Dim currentBalance As Double

        inputtedValue = CStr(Format(Me.bidInput.value, "General Number"))
        latestBid = CStr(Format(Me.currentBid.Caption, "General Number"))
        currentBalance = CStr(Format(Me.currentBalanceLabel.Caption, "General Number"))

        If inputtedValue < latestBid Then
            ' Lower than highest bid
            MsgBox "Please Enter a higher bid"
        Else
            If inputtedValue < currentBalance Then
                ' Value within price range of account (able to be done)
```

```
                Dim SQL As String
                Me.bidInput.SetFocus
                SQL = "INSERT INTO bidTable (userID, itemID, price, bidTime) " & _
                  "VALUES (" & Me.userID.Caption & ", " & Me.itemListBox.Column(0) & ", " & CStr(Fo
rmat(Me.bidInput.Text, "General Number")) & ", '" & Now() & "');"

                CurrentDb.Execute SQL

                Me.bidInput.value = ""
                Me.placeBid.SetFocus

                MsgBox "Bid successfully placed"
            Else
                MsgBox "Too much money, can't afford"
            End If
        End If
    End If
End Sub

Private Sub searchBox_Change()
    Dim vSearchString As String

    vSearchString = searchBox.Text

    searchHiddenInput.value = vSearchString

    Me.itemListBox.Requery

    If Len(Me.searchHiddenInput) <> 0 And InStr(Len(searchHiddenInput), searchHiddenInput, " ", vbT
extCompare) Then
        Exit Sub
    End If

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True

    updateItems

    Me.searchBox.SetFocus

    If Not IsNull(Len(Me.searchBox)) Then
        Me.searchBox.SelStart = Len(Me.searchBox)
    End If



End Sub
```

Form_registerForm - 1

```
Option Compare Database

Private Sub Form_Load()
    DoCmd.GoToRecord , , acNewRec
End Sub

Private Sub registerButton_Click()
    DoCmd.Close acForm, "registerForm", acSaveYes
End Sub
```

```
Form_sellForm - 1

Option Compare Database

Private Sub backButton_Click()
    DoCmd.Close acForm, "sellForm", acSaveYes
    DoCmd.OpenForm "mainForm"
End Sub

Private Sub fileSelection_Click()
    Dim fileDialog As Object
    Set fileDialog = Application.fileDialog(msoFileDialogOpen)
    With fileDialog
        .AllowMultiSelect = False
        .Title = "Please Select Item Image"
        .Filters.Clear
        .Filters.Add "Images", "*.png; *.jpg; *.jpeg", 1
    End With

    If fileDialog.Show Then
        fileDir = fileDialog.SelectedItems(1)

        ' Set to picturePath
        Me.fullPath.Caption = fileDir

        Dim fileName As String
        fileName = Split(fileDir, "\")(UBound(Split(fileDir, "\")))

        ' Rename button to filename
        Me.fileSelection.Caption = fileName
    End If
End Sub

Private Sub Form_Load()
    ' hide respective Labels
    Me.fullPath.Visible = False
    Me.currentUserID.Visible = False

    Me.fullPath.Caption = ""
    Me.fileSelection.Caption = "Select Image..."

    DoCmd.OpenForm "loginForm", , , , , acHidden

    Dim SQL As String
    SQL = "SELECT * from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"
    ' Populate Currency
    Me.currentBalanceLabel.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("currentBal
ance").value, 2)
    ' Populate ID
    Me.currentUserID.value = CurrentDb.OpenRecordset(SQL).Fields("userID").value
    ' Populate Username
    Me.loggedInUser.Caption = CurrentDb.OpenRecordset(SQL).Fields("firstName").value + " " + Curren
tDb.OpenRecordset(SQL).Fields("lastName").value

    Forms![loginForm].Visible = False
    Me.sellingItemsList.SetFocus
    Me.sellingItemsList.Text = "New Product"
    Me.bidInput.SetFocus
End Sub

Function GetFilenameFromPath(ByVal strPath As String) As String
' Returns the rightmost characters of a string upto but not including the rightmost '\'
' e.g. 'c:\winnt\win.ini' returns 'win.ini'

    If Right$(strPath, 1) <> "\" And Len(strPath) > 0 Then
        GetFilenameFromPath = GetFilenameFromPath(Left$(strPath, Len(strPath) - 1)) + Right$(strPat
h, 1)
    End If
End Function

Private Sub sellingItemsList_Change()
On Error GoTo errSellingItemsList
    If sellingItemsList.Text = "New Product" Then
        ' Wipe each field
        Me.bidInput.SetFocus
        Me.bidInput.Text = ""
```

```
        Me.bidDescription.SetFocus
        Me.bidDescription.Text = ""
        Me.fullPath.Caption = ""
        Me.fileSelection.Caption = "Select Image..."
        Me.bidStarting.SetFocus
        Me.bidStarting.Text = ""
        Me.endBidTime.SetFocus
        Me.endBidTime.Text = ""
        Me.bidInput.SetFocus
    Else
        ' Not the default field, change to respective values
        ' Execute SQL Query
        Dim SQL As String
        SQL = "Select * FROM itemEntity WHERE itemID=" + Me.sellingItemsList.value
        ' Populate Fields from query
        Me.bidInput.SetFocus
        Me.bidInput.Text = CurrentDb.OpenRecordset(SQL).Fields("itemName").value
        Me.bidDescription.SetFocus
        Me.bidDescription.Text = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").value
        Me.fullPath.Caption = CurrentDb.OpenRecordset(SQL).Fields("picture").value

        Me.fileSelection.Caption = GetFilenameFromPath(CurrentDb.OpenRecordset(SQL).Fields("picture
").value)

        Me.bidStarting.SetFocus
        Me.bidStarting.Text = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("initBid").value,
2)
        Me.endBidTime.SetFocus
        Me.endBidTime.Text = Format(CurrentDb.OpenRecordset(SQL).Fields("endBidtime").value, "dd/mm
/yyyy hh:mm:ss am/pm")
        Me.bidInput.SetFocus
    End If
errSellingItemsList:
    Exit Sub
End Sub

Private Sub submitButton_Click()
    Dim SQL As String
    Me.sellingItemsList.SetFocus
    If sellingItemsList.Text = "New Product" Then
        SQL = "INSERT INTO itemEntity (itemName, itemDescription, sellerID, picture, initBidTime, i
nitBid, endBidTime) " & _
          "VALUES ('" & Me.bidInput.value & "', '" & Me.bidDescription.value & "', " & Me.currentUs
erID.value & _
          ", '" & Me.fullPath.Caption & "' ,'" & Now() & "' ," & Me.bidStarting.value & " ,'" & Me.
endBidTime.value & "');"

          CurrentDb.Execute SQL

          MsgBox "Successfully Added Item"
    Else
        SQL = "UPDATE itemEntity " & _
              "SET itemName='" + Me.bidInput.value + "', itemDescription='" + Me.bidDescription.val
ue + "', " & _
                  "sellerID='" + CStr(Me.currentUserID.value) + "', picture='" + CStr(Me.fullPath.C
aption) + "', " & _
                  "initBidTime='" + CStr(Now()) + "', initBid='" + CStr(Me.bidStarting.value) + "',
 " & _
                  "endBidTime='" + Me.endBidTime.value + "' " & _
              "WHERE itemName='" + sellingItemsList.Text + "';"
          CurrentDb.Execute SQL
          MsgBox "Successfully Appended Item"
    End If
    Me.sellingItemsList.Requery
    Me.sellingItemsList.SetFocus
    Me.sellingItemsList.Text = Me.bidInput.value
    Me.submitButton.SetFocus
End Sub
```

Form_soldForm - 1

```
Option Compare Database

Public Sub VerticalAlignCenter(ByRef ctl As Control)
    Dim MinimumMargin As Integer
    Dim BorderWidth As Integer
    Dim TwipsPerPoint

    TwipsPerPoint = 20
    If Not ((TypeOf ctl Is TextBox) Or (TypeOf ctl Is Label)) Then Exit Sub

    'Figure out how many lines it is
    Dim LenOfText, WidOfBox, NumberOfLines, HtOfText
    If TypeOf ctl Is TextBox Then
        LenOfText = ctl.Text
    Else:
        LenOfText = ctl.Caption
    End If

    WidOfBox = ctl.Width
    LenOfText = (Len(LenOfText) * TwipsPerPoint * ctl.FontSize) / 2
    NumberOfLines = Int(LenOfText / WidOfBox) + 1
    HtOfText = NumberOfLines * TwipsPerPoint * ctl.FontSize


    MinimumMargin = 1 * TwipsPerPoint
    BorderWidth = (ctl.BorderWidth * TwipsPerPoint) / 2

    ctl.TopMargin = ((ctl.Height - HtOfText) / 2) - MinimumMargin - BorderWidth
End Sub

Private Sub backButton_Click()
    DoCmd.Close acForm, "soldForm", acSaveYes
    DoCmd.OpenForm "mainForm"
End Sub

Private Sub Form_Load()
    Me.itemDescription.Visible = False
    Me.pictureBox.Visible = True

    DoCmd.OpenForm "loginForm", , , , , acHidden
    Dim SQL As String

    SQL = "SELECT * from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    Me.currentBalanceLabel.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("currentBal
ance").value, 2)
    Forms![loginForm].Visible = False
    'CurrentDb.OpenRecordset(SQL, dbOpenSnapshot, dbReadOnly)
    ' Used to manipulate data faster, as it is only a ReadOnly and 'snapshot'
    ' Concatenate (bring multiple pieces together)
    ' Populate Username
    Me.loggedInUser.Caption = CurrentDb.OpenRecordset(SQL).Fields("firstName").value + " " + Curren
tDb.OpenRecordset(SQL, dbOpenSnapshot, dbReadOnly).Fields("lastName").value
    Me.userID.value = CurrentDb.OpenRecordset(SQL).Fields("userID").value
    VerticalAlignCenter Me.itemDescription

    Me.itemListBox.Requery

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True
End Sub

Private Sub Form_Timer()
    If IsNull(Me.itemListBox.Column(0)) Then
        ' Do nothing
        'Me.bidLength.Caption = "-"
    Else
        Dim SQL As String
        Dim bidSQL As String

        SQL = "SELECT soldItems.*, itemEntity.* " & _
          "FROM itemEntity INNER JOIN soldItems ON itemEntity.itemID = soldItems.itemID " & _
          "WHERE soldItems.itemID = " & Me.itemListBox.Column(0)
```

```
        ' Update Latest bid BEFORE updating timing interval
        If Me.currentBid.Caption = "currentPrice" Then
            ' Set item description
            Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").val
ue
            ' Set image
            Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
            ' Set price
            Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("soldPrice")
.value, 2)
        Else
            ' Do nothing
        End If
    End If
End Sub

Private Sub itemDescription_Click()
    Me.itemDescription.Visible = False
    Me.pictureBox.Visible = True
End Sub
Private Sub updateItems()
On Error GoTo errUpdateItems
    ' Now that row value has been found (representing itemID in itemEntityTable),
    ' Call SQL query to populate other fields.
    Dim SQL As String
    Dim bidSQL As String

    SQL = "SELECT soldItems.*, itemEntity.* " & _
          "FROM itemEntity INNER JOIN soldItems ON itemEntity.itemID = soldItems.itemID " & _
          "WHERE soldItems.itemID = " & Me.itemListBox.Column(0)
    ' NEED:
    ' Item Description
    ' Picture Location
    ' initBid
    ' endBidTime

    ' Set item description
    Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").value
    ' Set image
    Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
    ' Set latest Bid
    Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("soldPrice").value,
2)
errUpdateItems:
    Exit Sub
End Sub

Private Sub itemListBox_AfterUpdate()
    updateItems
End Sub

Private Sub pictureBox_Click()
    Me.itemDescription.Visible = True
    Me.pictureBox.Visible = False
End Sub

Private Sub searchBox_Change()
    Dim vSearchString As String

    vSearchString = searchBox.Text

    searchHiddenInput.value = vSearchString

    Me.itemListBox.Requery

    If Len(Me.searchHiddenInput) <> 0 And InStr(Len(searchHiddenInput), searchHiddenInput, " ", vbT
extCompare) Then
        Exit Sub
    End If

    Me.itemListBox = Me.itemListBox.ItemData(0)
    Me.itemListBox.SetFocus
    Me.itemListBox.Selected(0) = True

    updateItems
```

```
    Me.searchBox.SetFocus

    If Not IsNull(Len(Me.searchBox)) Then
        Me.searchBox.SelStart = Len(Me.searchBox)
    End If



End Sub
```

Form_userProfileForm - 1

```
Option Compare Database

Private Sub Form_Load()
    DoCmd.OpenForm "loginForm", , , , , acHidden

    Dim SQL As String
    SQL = "SELECT userID from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    DoCmd.GoToRecord acDataForm, Me.Name, acGoTo, CInt(CurrentDb.OpenRecordset(SQL).Fields("userID"
).value)

    Forms![loginForm].Visible = False
End Sub

Private Sub updateButton_Click()
    DoCmd.Close acForm, "userProfileForm", acSaveYes
End Sub
```