**Case Two**

**An Income Tax Return**

**2 April 2015**

****

**Ricky Genz**

Table of Contents

[Part A – Test Data 3](#_Toc415679559)

[Set One – Edward Ellington 4](#_Toc415679560)

[11](#_Toc415679561)

[Set Two – Frank and Fran Falcon 12](#_Toc415679562)

[Set Three – Geraldine Gardner 21](#_Toc415679563)

[Part B – Technical Documentation 28](#_Toc415679564)

[Module 29](#_Toc415679565)

[About Form 30](#_Toc415679566)

[Main Form 31](#_Toc415679567)

[Schedule A 34](#_Toc415679568)

[Child Care Worksheet 37](#_Toc415679570)

[Schedule B 39](#_Toc415679571)

[Schedule C 43](#_Toc415679572)

[Schedule D 45](#_Toc415679573)

[Schedule E 49](#_Toc415679574)

[Form 1040 53](#_Toc415679575)

[Credit Card Form 59](#_Toc415679576)

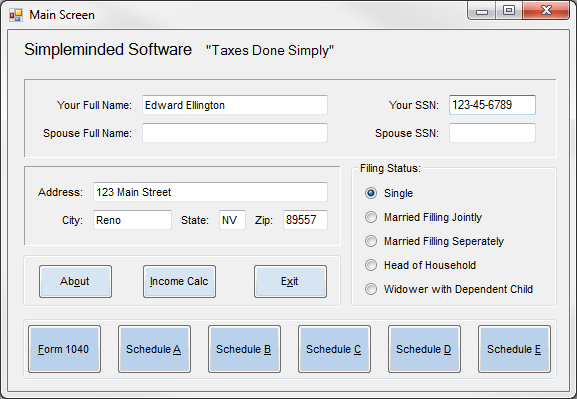
[Part C – Enhancements 65](#_Toc415679577)

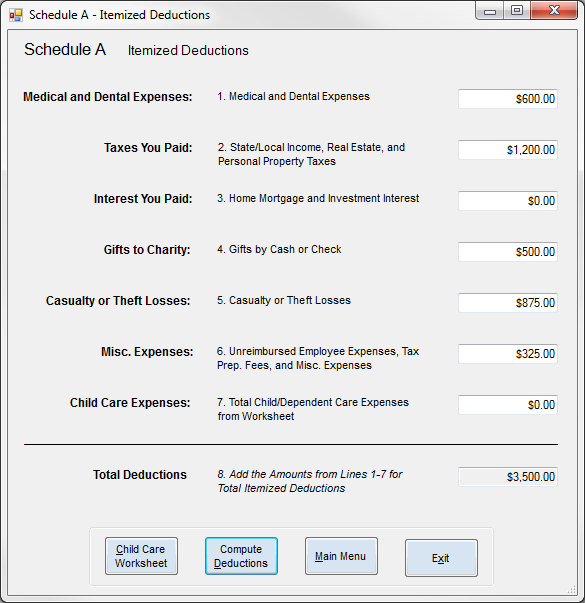
[Enhancement 1 – Zip Code Validation 66](#_Toc415679578)

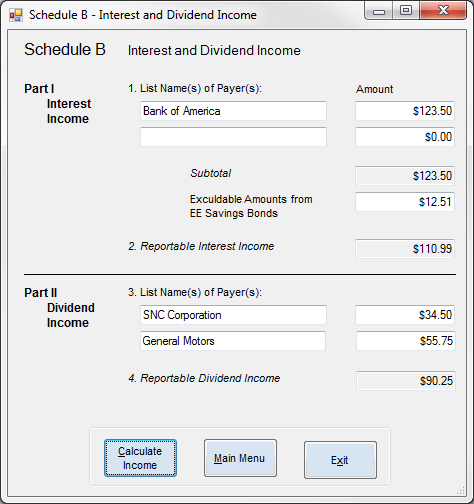
[Enhancement 2 – About Form 67](#_Toc415679579)

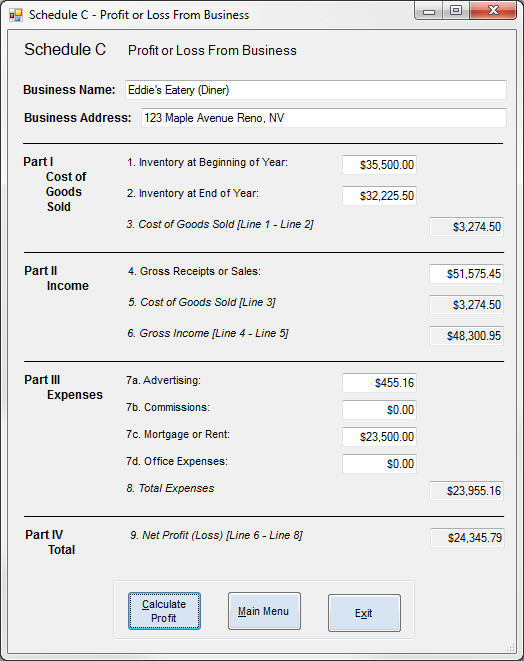
# Part A – Test Data

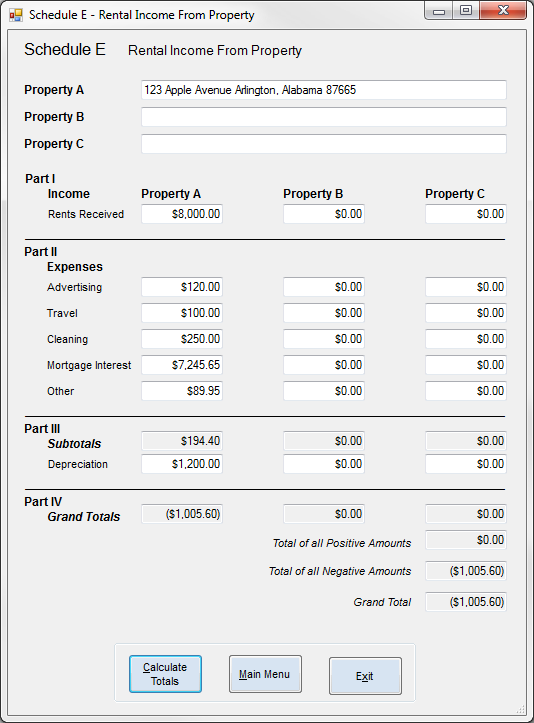
## Set One – Edward Ellington

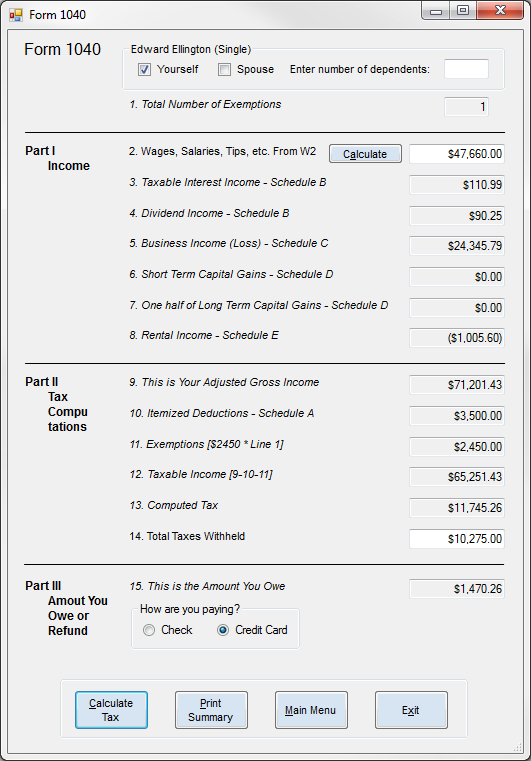


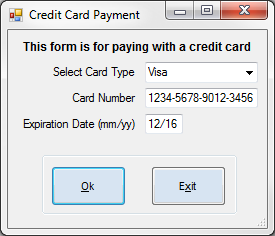
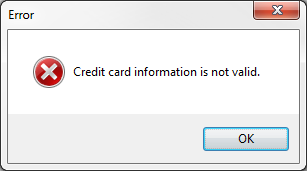






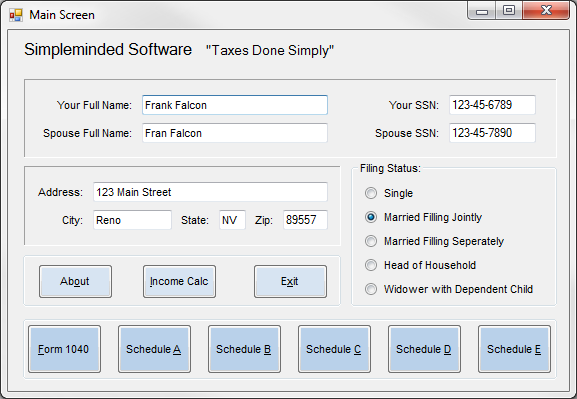


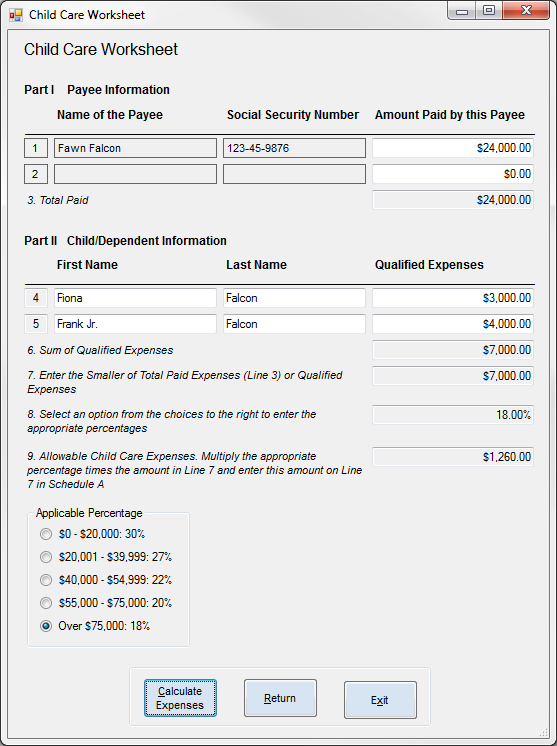


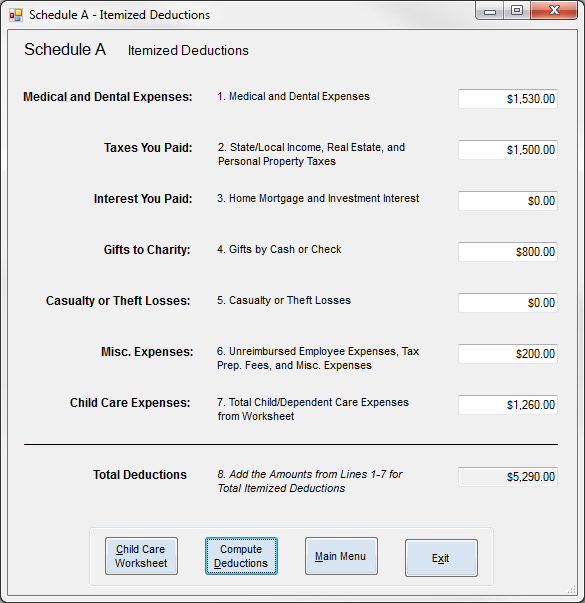


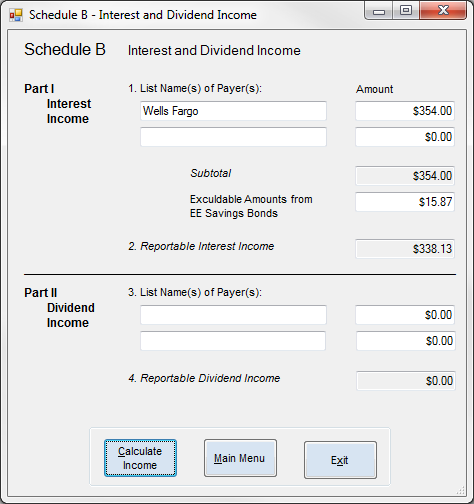
## 

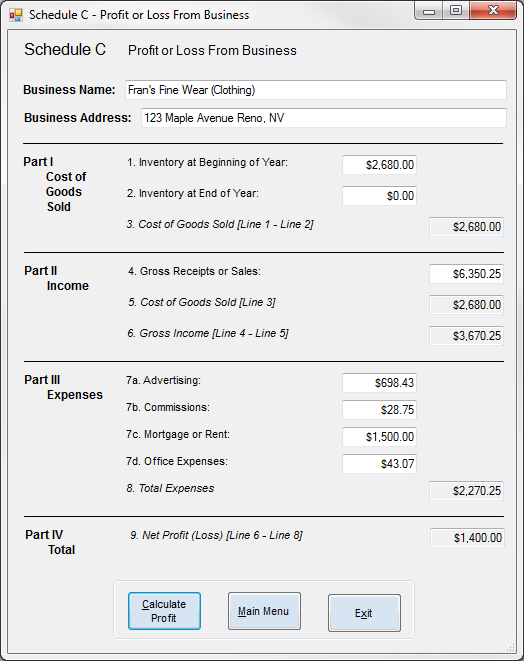
## Set Two – Frank and Fran Falcon

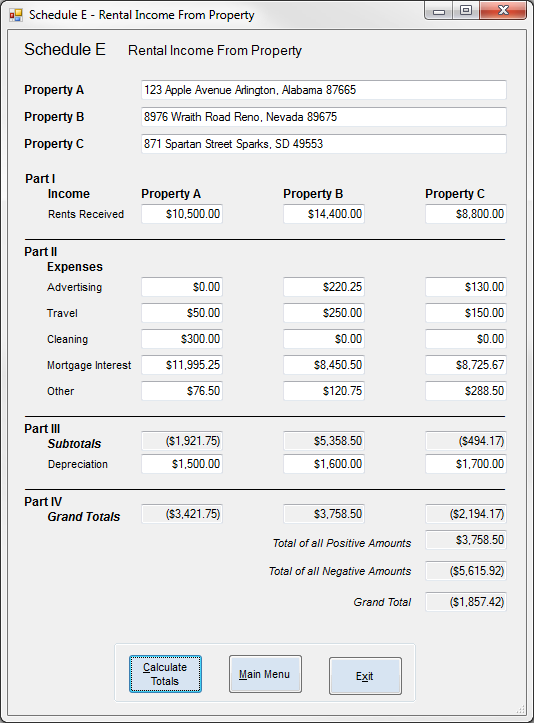


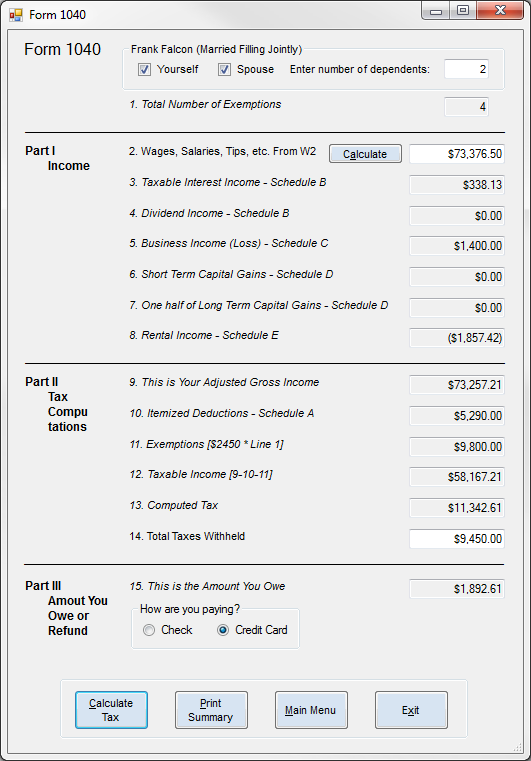


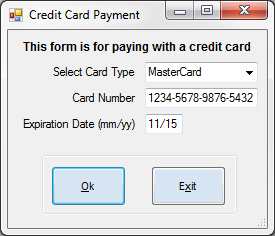


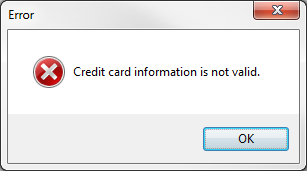


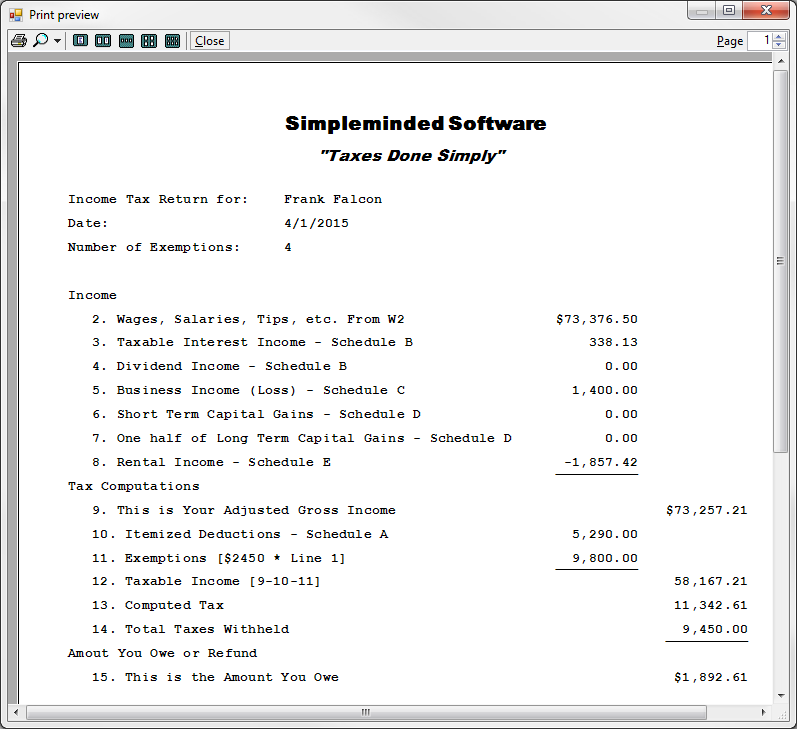




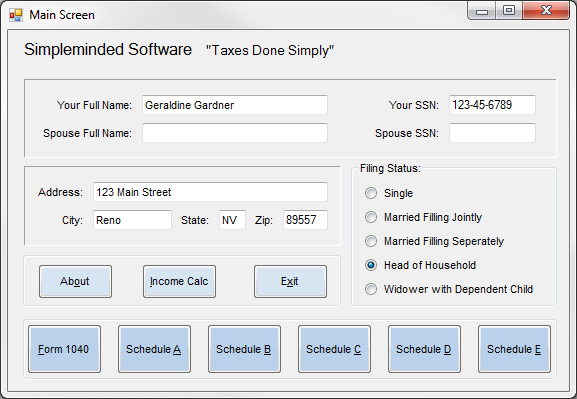


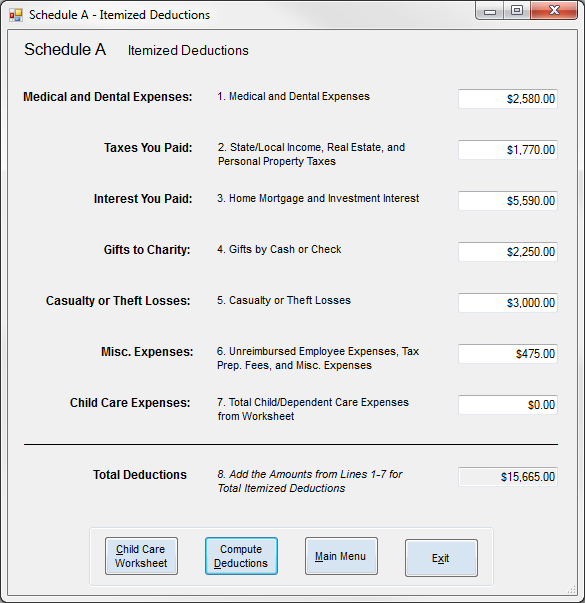


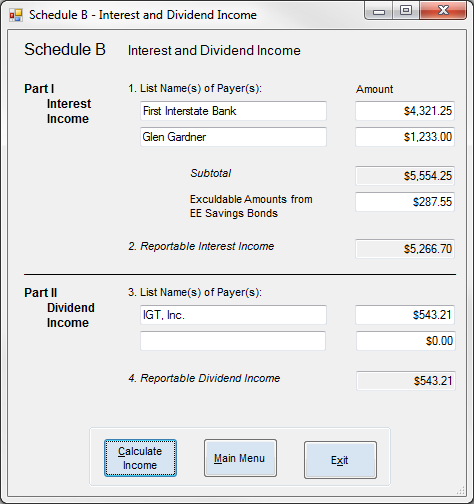


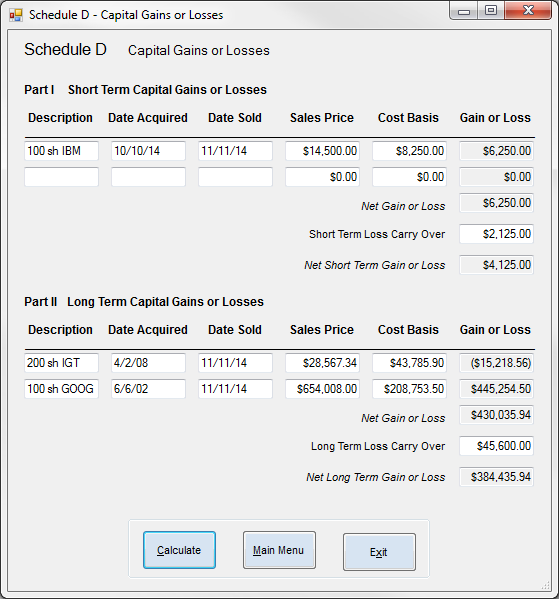


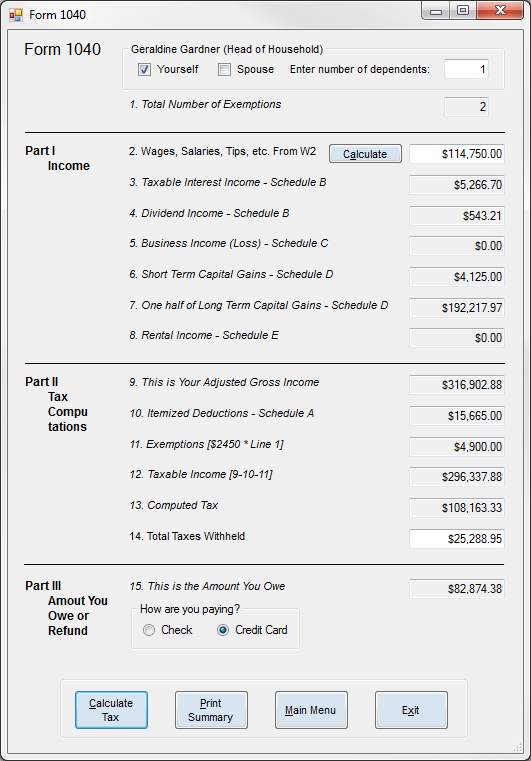
## Set Three – Geraldine Gardner

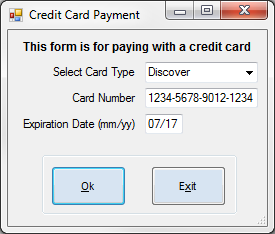
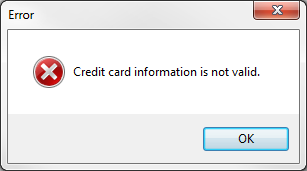


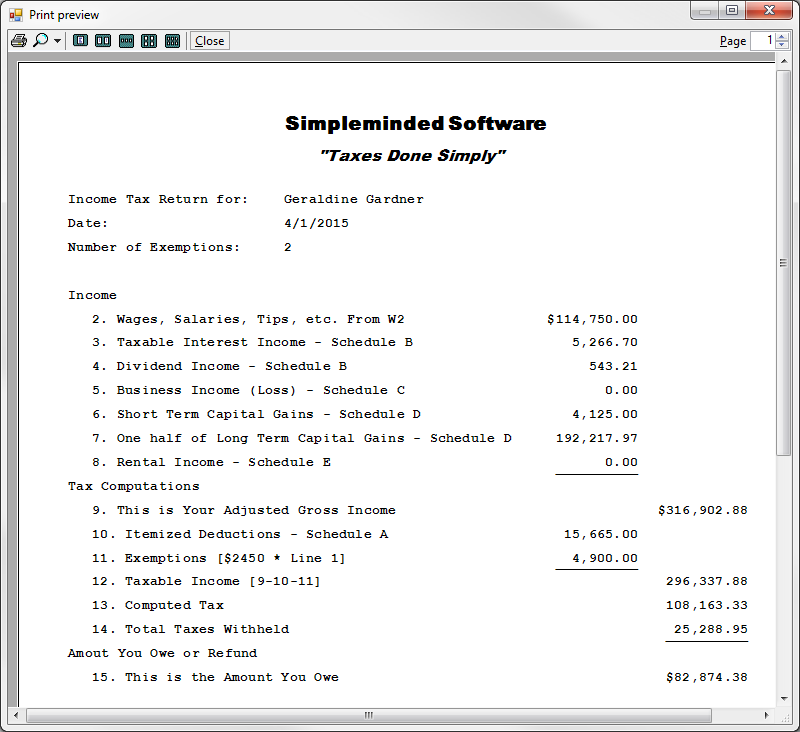












# Part B – Technical Documentation

## Module

Module modMain

'Global variable declaration

Public Title, Slogan, YName, FSString As String

Public FSNum, SATotal, CCWTotal, IITotal, DITotal, SCTotal, STTotal, LTTotal, SETotal As Decimal

Public Exemptions, Income, AGI, TotalExemptions, TaxableIncome, TotalTax, Withheld, Taxes As Decimal

Sub IsInvalidSSN(ByRef o As Object, ByRef b As Boolean)

'Tests for invalid Social Security Number

If Not o.Text Like "###-##-####" Then

MsgBox(o.AccessibleName & " is not valid. Must be in ###-##-#### format.", MsgBoxStyle.Critical, "Error")

o.Focus()

b = True

Else

b = False

End If

End Sub

Function IsBlank(o As Object) As Boolean

'Tests for blank input

If (Len(Trim(o.Text.ToString)) > 0) Then Return False

Return True

End Function

Function IsNegative(d As Decimal) As Boolean

'Tests for negative input

If (d >= 0) Then Return False

Return True

End Function

Function CZer(o As Object) As Decimal

'Converts textbox from blank or non-numeric to zero

If o = "" Then o = 0

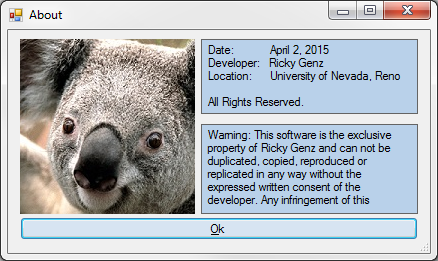
If Not IsNumeric(o) Then o = 0

Return o

End Function

End Module

## About Form



Public Class frmAbout

Private Sub OK(sender As Object, e As EventArgs) Handles btnOk.Click

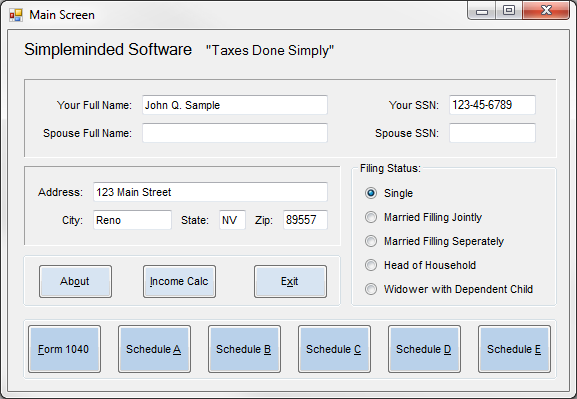
'Returns to the main form

Close()

End Sub

End Class

## Main Form



Public Class frmMain

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

If Len(Trim(c.Text)) <= 0 Then

MsgBox(c.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

End If

If c.Name = "txtYSSN" Then

Dim b As Boolean

Call IsInvalidSSN(c, b)

If b Then Return True

ElseIf c.Name = "txtSName" Then

Dim b As Boolean

If IsBlank(c) = False Then Call IsInvalidSSN(txtSSSN, b)

If b Then Return True

ElseIf c.Name = "txtZip" Then

If Not c.Text Like "#####" Then

MsgBox(c.AccessibleName & " is not valid. Must be in ##### format.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnAbout.Click, btnExit.Click

'Opens the about form

If sender Is btnAbout Then frmAbout.ShowDialog()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub FormLoad(sender As Object, e As EventArgs) Handles MyBase.Load

Title = lblTitle.Text

Slogan = lblSlogan.Text

End Sub

Private Sub OpenForms(sender As Object, e As EventArgs) Handles btnIncome.Click, btnF1040.Click, btnSA.Click, btnSB.Click, btnSC.Click, btnSD.Click, btnSE.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Stores name and filling status in global variables

YName = txtYName.Text

If rdoSingle.Checked Then FSNum = 1 : FSString = "Single"

If rdoJointly.Checked Then FSNum = 2 : FSString = "Married Filling Jointly"

If rdoSeperately.Checked Then FSNum = 1 : FSString = "Married Filling Seperately"

If rdoHoH.Checked Then FSNum = 3 : FSString = "Head of Household"

If rdoWidow.Checked Then FSNum = 3 : FSString = "Widower with Dependent Child"

'Opens each additional form depending on which button is clicked

If sender Is btnIncome Then frmIncome.ShowDialog()

If sender Is btnF1040 Then frm1040.ShowDialog()

If sender Is btnSA Then frmSA.ShowDialog()

If sender Is btnSB Then frmSB.ShowDialog()

If sender Is btnSC Then frmSC.ShowDialog()

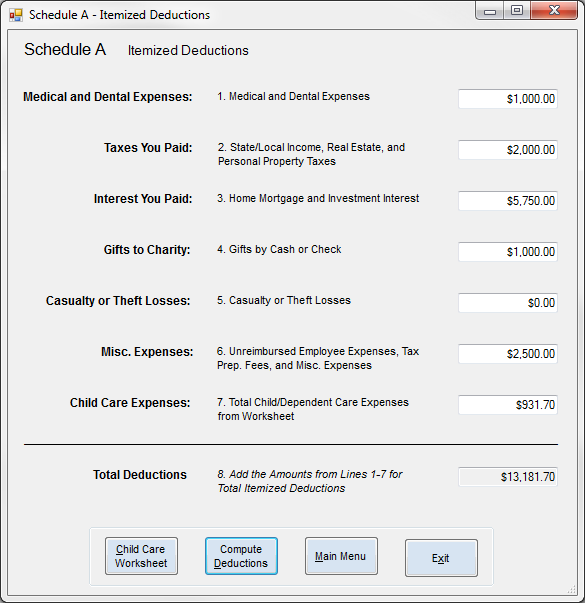
If sender Is btnSD Then frmSD.ShowDialog()

If sender Is btnSE Then frmSE.ShowDialog()

End Sub

End Class

## Schedule A



## 

Public Class frmSA

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnCCW.Click, btnReturn.Click, btnExit.Click

'Opens the child care worksheet form, saves the total in a global variable, and then reruns the Calculate sub

If sender Is btnCCW Then

frmCCW.ShowDialog()

txtCCExp.Text = CCWTotal

Calculate(sender, e)

End If

'Returns to the main form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCompute.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Stores amounts in local variables and formats as currency

Dim L1 = CZer(txtMDExpenses.Text)

txtMDExpenses.Text = FormatCurrency(L1)

Dim L2 = CZer(txtTaxesPaid.Text)

txtTaxesPaid.Text = FormatCurrency(L2)

Dim L3 = CZer(txtInterestPaid.Text)

txtInterestPaid.Text = FormatCurrency(L3)

Dim L4 = CZer(txtCharity.Text)

txtCharity.Text = FormatCurrency(L4)

Dim L5 = CZer(txtLosses.Text)

txtLosses.Text = FormatCurrency(L5)

Dim L6 = CZer(txtMiscExp.Text)

txtMiscExp.Text = FormatCurrency(L6)

Dim L7 = CZer(txtCCExp.Text)

txtCCExp.Text = FormatCurrency(L7)

'Calculates the total, saves in global variable, and then formats as currency

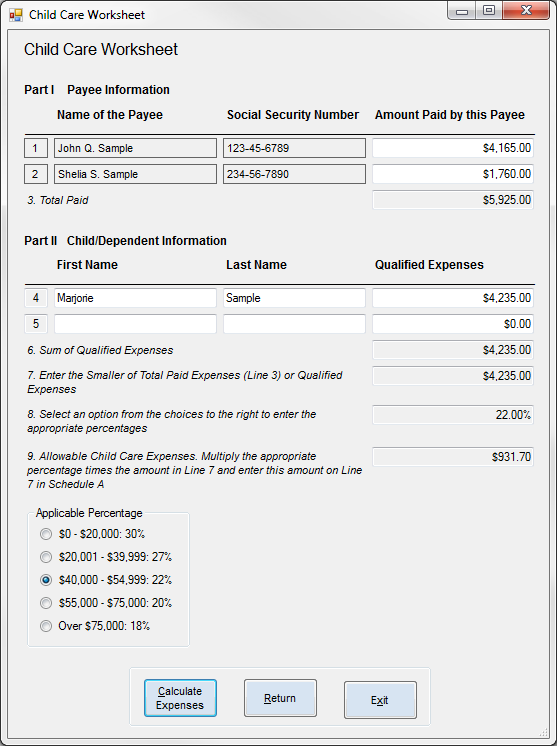
SATotal = L1 + L2 + L3 + L4 + L5 + L6 + L7

txtTotal.Text = FormatCurrency(SATotal)

End Sub

End Class

### Child Care Worksheet



Public Class frmCCW

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnReturn.Click, btnExit.Click

'Returns to Schedule A

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub PayeeOne(sender As Object, e As EventArgs) Handles lblPayeeOne.Click

'Populates Payee One information on Payee One label click

Dim PayeeOneName = InputBox("Please enter the name of the first payee.", "Name of Payee One")

Dim PayeeOneSSN = InputBox("Please enter the social security number of the first payee.", "SSN of Payee One")

lblPayeeOneName.Text = PayeeOneName

lblPayeeOneSSN.Text = PayeeOneSSN

End Sub

Private Sub PayeeTwo(sender As Object, e As EventArgs) Handles lblPayeeTwo.Click

'Populates Payee Two information on Payee Two label click

Dim PayeeTwoName = InputBox("Please enter the name of the second payee.", "Name of Payee Two")

Dim PayeeTwoSSN = InputBox("Please enter the social security number of the second payee.", "SSN of Payee Two")

lblPayeeTwoName.Text = PayeeTwoName

lblPayeeTwoSSN.Text = PayeeTwoSSN

End Sub

## Schedule B

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCalculate.Click

'Runs invalid SSN test if Payee One or Payee Two has information present

Dim b As Boolean = False

If IsBlank(lblPayeeOneName) = False Then Call IsInvalidSSN(lblPayeeOneSSN, b)

If b Then Exit Sub

If IsBlank(lblPayeeTwoName) = False Then Call IsInvalidSSN(lblPayeeTwoSSN, b)

If b Then Exit Sub

'Stores amounts in local variables and formats as currency

Dim L1 = CZer(txtPayeeOnePaid.Text)

txtPayeeOnePaid.Text = FormatCurrency(L1)

Dim L2 = CZer(txtPayeeTwoPaid.Text)

txtPayeeTwoPaid.Text = FormatCurrency(L2)

Dim L3 = L1 + L2

txtTotalPaid.Text = FormatCurrency(L3)

Dim L4 = CZer(txtExpFour.Text)

txtExpFour.Text = FormatCurrency(L4)

Dim L5 = CZer(txtExpFive.Text)

txtExpFive.Text = FormatCurrency(L5)

Dim L6 = L4 + L5

txtExpSum.Text = FormatCurrency(L6)

'Tests both subtotals to see which is smaller

Dim L7

If L3 < L6 Then L7 = L3 Else L7 = L6

txtSmaller.Text = FormatCurrency(L7)

'Stores the selected percentage in a local variable

Dim L8 = 0.0

If rdo30.Checked Then L8 = 0.3

If rdo27.Checked Then L8 = 0.27

If rdo22.Checked Then L8 = 0.22

If rdo20.Checked Then L8 = 0.2

If rdo18.Checked Then L8 = 0.18

txtPercentage.Text = FormatPercent(L8)

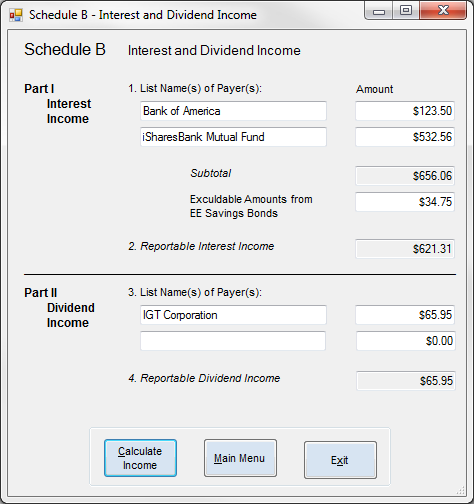
'Calculates the total, saves in global variable, and then formats as currency

CCWTotal = L8 \* L7

txtCCExp.Text = FormatCurrency(CCWTotal)

End Sub

End Class



Public Class frmSB

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnReturn.Click, btnExit.Click

'Returns to the main form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCalculate.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Stores amounts in local variables and formats as currency

Dim L1a = CZer(txtInterestAmountOne.Text)

txtInterestAmountOne.Text = FormatCurrency(L1a)

Dim L1b = CZer(txtInterestAmountTwo.Text)

txtInterestAmountTwo.Text = FormatCurrency(L1b)

Dim L1c = L1a + L1b

txtInterestSubtotal.Text = FormatCurrency(L1c)

Dim L1d = CZer(txtInterestExcludable.Text)

txtInterestExcludable.Text = FormatCurrency(L1d)

'Calculates the total, saves in global variable, and then formats as currency

IITotal = L1c - L1d

txtInterestTotal.Text = FormatCurrency(IITotal)

'Stores amounts in local variables and formats as currency

Dim L3a = CZer(txtDividendAmountOne.Text)

txtDividendAmountOne.Text = FormatCurrency(L3a)

Dim L3b = CZer(txtDividendAmountTwo.Text)

txtDividendAmountTwo.Text = FormatCurrency(L3b)

'Calculates the total, saves in global variable, and then formats as currency

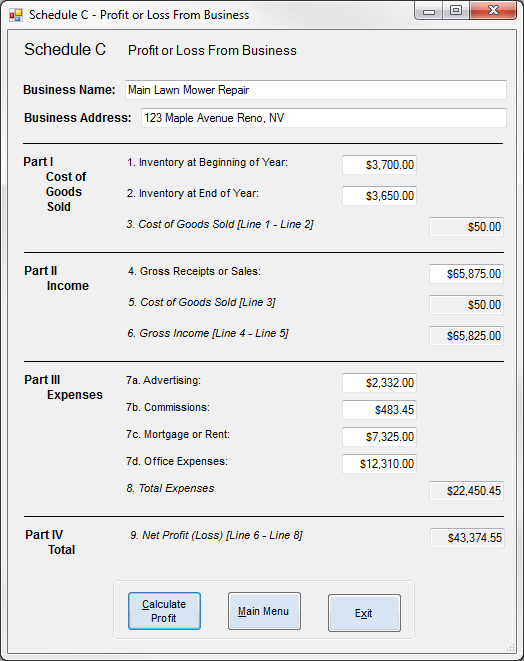
DITotal = L3a + L3b

txtDividendTotal.Text = FormatCurrency(DITotal)

End Sub

End Class

## Schedule C



Public Class frmSC

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

If c.Name = "txtName" Then

If IsBlank(c) Then

MsgBox(c.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

ElseIf c.Name = "txtAddress" Then

If IsBlank(c) Then

MsgBox(c.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnReturn.Click, btnExit.Click

'Returns to the main form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

## Schedule D

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCalculate.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Stores amounts in local variables and formats as currency

Dim L1 = CZer(txtInvBeg.Text)

txtInvBeg.Text = FormatCurrency(L1)

Dim L2 = CZer(txtInvEnd.Text)

txtInvEnd.Text = FormatCurrency(L2)

Dim L3 = L1 - L2

If L3 < 0 Then L3 = 0

txtCOGS.Text = FormatCurrency(L3)

Dim L4 = CZer(txtSales.Text)

txtSales.Text = FormatCurrency(L4)

txtCOGS5.Text = FormatCurrency(L3)

Dim L6 = L4 - L3

txtGrossIncome.Text = FormatCurrency(L6)

Dim L7a = CZer(txtAdvertising.Text)

txtAdvertising.Text = FormatCurrency(L7a)

Dim L7b = CZer(txtCommissions.Text)

txtCommissions.Text = FormatCurrency(L7b)

Dim L7c = CZer(txtMortgage.Text)

txtMortgage.Text = FormatCurrency(L7c)

Dim L7d = CZer(txtOffice.Text)

txtOffice.Text = FormatCurrency(L7d)

Dim L8 = L7a + L7b + L7c + L7d

txtTotal.Text = FormatCurrency(L8)

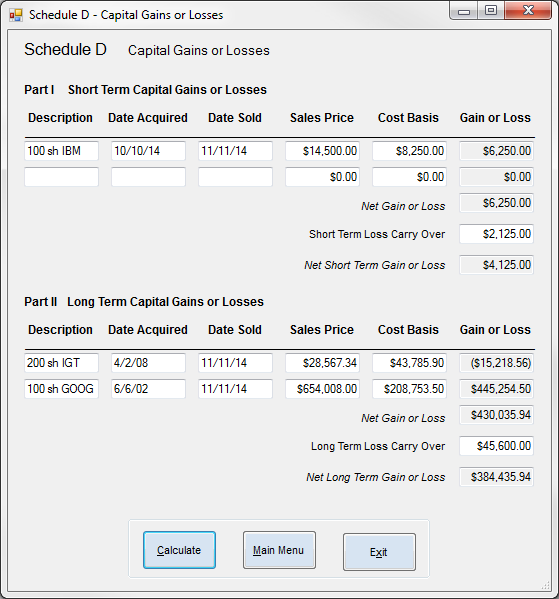
'Calculates the total, saves in global variable, and then formats as currency

SCTotal = L6 - L8

txtNetProfit.Text = FormatCurrency(SCTotal)

End Sub

End Class



Public Class frmSD

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnReturn.Click, btnExit.Click

'Returns to the main form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCalculate.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Stores amounts in local variables and formats as currency

Dim STP1 = CZer(txtSTPriceOne.Text)

txtSTPriceOne.Text = FormatCurrency(STP1)

Dim STC1 = CZer(txtSTCostOne.Text)

txtSTCostOne.Text = FormatCurrency(STC1)

Dim STGL1 = STP1 - STC1

txtSTGLOne.Text = FormatCurrency(STGL1)

Dim STP2 = CZer(txtSTPriceTwo.Text)

txtSTPriceTwo.Text = FormatCurrency(STP2)

Dim STC2 = CZer(txtSTCostTwo.Text)

txtSTCostTwo.Text = FormatCurrency(STC2)

Dim STGL2 = STP2 - STC2

txtSTGLTwo.Text = FormatCurrency(STGL2)

'Calculates the total, saves in global variable, and then formats as currency

Dim STGL = STGL1 + STGL2

txtSTNetGL.Text = FormatCurrency(STGL)

Dim STCO = CZer(txtSTCarryOver.Text)

txtSTCarryOver.Text = FormatCurrency(STCO)

STTotal = STGL - STCO

txtNetSTGL.Text = FormatCurrency(STTotal)

'Stores amounts in local variables and formats as currency

Dim LTP1 = CZer(txtLTPriceOne.Text)

txtLTPriceOne.Text = FormatCurrency(LTP1)

Dim LTC1 = CZer(txtLTCostOne.Text)

txtLTCostOne.Text = FormatCurrency(LTC1)

Dim LTGL1 = LTP1 - LTC1

txtLTGLOne.Text = FormatCurrency(LTGL1)

Dim LTP2 = CZer(txtLTPriceTwo.Text)

txtLTPriceTwo.Text = FormatCurrency(LTP2)

Dim LTC2 = CZer(txtLTCostTwo.Text)

txtLTCostTwo.Text = FormatCurrency(LTC2)

Dim LTGL2 = LTP2 - LTC2

txtLTGLTwo.Text = FormatCurrency(LTGL2)

'Calculates the total, saves in global variable, and then formats as currency

Dim LTGL = LTGL1 + LTGL2

txtLTNetGL.Text = FormatCurrency(LTGL)

Dim LTCO = CZer(txtLTCarryOver.Text)

txtLTCarryOver.Text = FormatCurrency(LTCO)

LTTotal = LTGL - LTCO

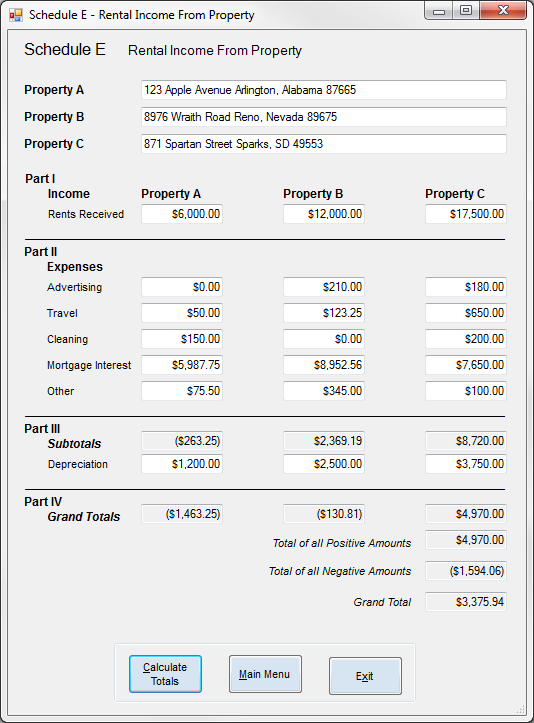
txtNetLTGL.Text = FormatCurrency(LTTotal)

LTTotal = LTTotal / 2

End Sub

End Class

## Schedule E



Public Class frmSE

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

If c.Name = "txtAName" Then

If Not IsBlank(c) Then

If IsBlank(txtARent) Then

MsgBox(txtARent.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

End If

ElseIf c.Name = "txtBName" Then

If Not IsBlank(c) Then

If IsBlank(txtBRent) Then

MsgBox(txtBRent.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

End If

ElseIf c.Name = "txtCName" Then

If Not IsBlank(c) Then

If IsBlank(txtCRent) Then

MsgBox(txtCRent.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnReturn.Click, btnExit.Click

'Returns to the main form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCalculate.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'For Property A stores amounts in local variables and formats as currency

Dim AR = CZer(txtARent.Text)

txtARent.Text = FormatCurrency(AR)

Dim AA = CZer(txtAAds.Text)

txtAAds.Text = FormatCurrency(AA)

Dim AT = CZer(txtATravel.Text)

txtATravel.Text = FormatCurrency(AT)

Dim AC = CZer(txtACleaning.Text)

txtACleaning.Text = FormatCurrency(AC)

Dim AM = CZer(txtAInterest.Text)

txtAInterest.Text = FormatCurrency(AM)

Dim AO = CZer(txtAOther.Text)

txtAOther.Text = FormatCurrency(AO)

'For Property A calculates the total and then formats as currency

Dim AB = AR - (AA + AT + AC + AM + AO)

txtASub.Text = FormatCurrency(AB)

Dim AD = CZer(txtADep.Text)

txtADep.Text = FormatCurrency(AD)

Dim AG = AB - AD

txtATotal.Text = FormatCurrency(AG)

'For Property B stores amounts in local variables and formats as currency

Dim BR = CZer(txtBRent.Text)

txtBRent.Text = FormatCurrency(BR)

Dim BA = CZer(txtBAds.Text)

txtBAds.Text = FormatCurrency(BA)

Dim BT = CZer(txtBTravel.Text)

txtBTravel.Text = FormatCurrency(BT)

Dim BC = CZer(txtBCleaning.Text)

txtBCleaning.Text = FormatCurrency(BC)

Dim BM = CZer(txtBInterest.Text)

txtBInterest.Text = FormatCurrency(BM)

Dim BO = CZer(txtBOther.Text)

txtBOther.Text = FormatCurrency(BO)

'For Property B calculates the total and then formats as currency

Dim BB = BR - (BA + BT + BC + BM + BO)

txtBSub.Text = FormatCurrency(BB)

Dim BD = CZer(txtBDep.Text)

txtBDep.Text = FormatCurrency(BD)

Dim BG = BB - BD

txtBTotal.Text = FormatCurrency(BG)

'For Property C stores amounts in local variables and formats as currency

Dim CR = CZer(txtCRent.Text)

txtCRent.Text = FormatCurrency(CR)

Dim CA = CZer(txtCAds.Text)

txtCAds.Text = FormatCurrency(CA)

Dim CT = CZer(txtCTravel.Text)

txtCTravel.Text = FormatCurrency(CT)

Dim CC = CZer(txtCCleaning.Text)

txtCCleaning.Text = FormatCurrency(CC)

Dim CM = CZer(txtCInterest.Text)

txtCInterest.Text = FormatCurrency(CM)

Dim CO = CZer(txtCOther.Text)

txtCOther.Text = FormatCurrency(CO)

'For Property C calculates the total and then formats as currency

Dim CB = CR - (CA + CT + CC + CM + CO)

txtCSub.Text = FormatCurrency(CB)

Dim CD = CZer(txtCDep.Text)

txtCDep.Text = FormatCurrency(CD)

Dim CG = CB - CD

txtCTotal.Text = FormatCurrency(CG)

'Determines positive and negative total amounts

Dim PT, NT As Decimal

If AG > 0 Then PT += AG Else NT += AG

If BG > 0 Then PT += BG Else NT += BG

If CG > 0 Then PT += CG Else NT += CG

txtPositiveTotal.Text = FormatCurrency(PT)

txtNegativeTotal.Text = FormatCurrency(NT)

'Calculates the total, saves in global variable, and then formats as currency

SETotal = PT + NT

txtGrandTotal.Text = FormatCurrency(SETotal)

End Sub

End Class

## Form 1040

Public Class frm1040

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

If c.Name = "txtIncome" Then

Dim num = CZer(c.Text)

If num < 1 Or num > 500000 Then

MsgBox(c.AccessibleName & " is not valid. Must be between $1.00 and $500,000.00.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnIncome.Click, chkCreditCard.CheckedChanged, btnReturn.Click, btnExit.Click

'Opens the credit card form

If sender Is chkCreditCard Then frmCC.ShowDialog()

'Opens the wage calculation form

If sender Is btnIncome Then frmIncome.ShowDialog()

'Returns to the main form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub FormLoad(sender As Object, e As EventArgs) Handles MyBase.Load

'Initialize all neccessary values into the form from global variables

grpYou.Text = YName & " (" & FSString & ")"

txtIncome.Text = FormatCurrency(Income)

txtInterest.Text = FormatCurrency(IITotal)

txtDividend.Text = FormatCurrency(DITotal)

txtBusiness.Text = FormatCurrency(SCTotal)

txtSTGL.Text = FormatCurrency(STTotal)

txtLTGL.Text = FormatCurrency(LTTotal)

txtRental.Text = FormatCurrency(SETotal)

txtDeductions.Text = FormatCurrency(SATotal)

End Sub

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCalculate.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Calculates amounts, saves in global variables, and then formats as currency

Exemptions = CZer(txtDependents.Text)

If chkYourself.Checked Then Exemptions += 1

If chkSpouse.Checked Then Exemptions += 1

txtExemptionsNumber.Text = Format(Exemptions, 0)

Income = CZer(txtIncome.Text)

txtIncome.Text = FormatCurrency(Income)

AGI = Income + IITotal + DITotal + SCTotal + STTotal + LTTotal + SETotal

txtAGI.Text = FormatCurrency(AGI)

TotalExemptions = Exemptions \* 2450

txtExemptions.Text = FormatCurrency(TotalExemptions)

TaxableIncome = AGI - SATotal - TotalExemptions

txtTaxableIncome.Text = FormatCurrency(TaxableIncome)



Private Sub PrintPageSummary(sender As Object, e As Printing.PrintPageEventArgs) Handles PrintSummary.PrintPage

'Declare and set printing variables.

Dim g As Graphics = e.Graphics

Dim x0 As Single = 50

Dim x1 As Single = 550

Dim x2 As Single = 75

Dim x3 As Single = 665

Dim x4 As Single = 275

Dim x5 As Single = 275

Dim x6 As Single = 310

Dim y0 As Single = 50

Dim y1 As Single = 25

Dim y2 As Single = 50

Dim y3 As Single = 5

Dim y4 As Single = 35

Dim h1 As New Font("Arial Black", 14, FontStyle.Bold)

Dim h2 As New Font("Arial Black", 12, FontStyle.Italic)

Dim b As New Font("Courier New", 10, FontStyle.Bold)

Dim fmt As String = "{0,11:n2}"

Dim fmtc As String = "{0,11:c2}"

'Draw heading.

g.DrawString(Title, h1, Brushes.Black, x5, y0)

y0 += y4

g.DrawString(Slogan, h2, Brushes.Black, x6, y0)

y0 += y2

'Draw customer information.

g.DrawString("Income Tax Return for:", b, Brushes.Black, x0, y0)

g.DrawString(YName, b, Brushes.Black, x4, y0)

y0 += y1

g.DrawString("Date:", b, Brushes.Black, x0, y0)

g.DrawString(Today, b, Brushes.Black, x4, y0)

y0 += y1

g.DrawString("Number of Exemptions:", b, Brushes.Black, x0, y0)

g.DrawString(Exemptions, b, Brushes.Black, x4, y0)

y0 += y2

'Draw summary amounts.

g.DrawString(lblPartOneDesc.Text, b, Brushes.Black, x0, y0)

y0 += y1

g.DrawString(lblTwo.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmtc, Income), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblThree.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, IITotal), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblFour.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, DITotal), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblFive.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, SCTotal), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblSix.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, STTotal), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblSeven.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, LTTotal), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblEight.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, SETotal), b, Brushes.Black, x1, y0)

g.DrawString(String.Format(fmt, "\_\_\_\_\_\_\_\_\_\_"), b, Brushes.Black, x1, y0 + y3)

y0 += y1

g.DrawString(lblPartTwoDesc.Text, b, Brushes.Black, x0, y0)

y0 += y1

g.DrawString(lblNine.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmtc, AGI), b, Brushes.Black, x3, y0)

y0 += y1

g.DrawString(lblTen.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, SATotal), b, Brushes.Black, x1, y0)

y0 += y1

g.DrawString(lblEleven.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, TotalExemptions), b, Brushes.Black, x1, y0)

g.DrawString(String.Format(fmt, "\_\_\_\_\_\_\_\_\_\_"), b, Brushes.Black, x1, y0 + y3)

y0 += y1

g.DrawString(lblTwelve.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, TaxableIncome), b, Brushes.Black, x3, y0)

y0 += y1

g.DrawString(lblThirteen.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, TotalTax), b, Brushes.Black, x3, y0)

y0 += y1

g.DrawString(lblFourteen.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmt, Withheld), b, Brushes.Black, x3, y0)

g.DrawString(String.Format(fmt, "\_\_\_\_\_\_\_\_\_\_"), b, Brushes.Black, x3, y0 + y3)

y0 += y1

g.DrawString(lblPartThreeDesc.Text, b, Brushes.Black, x0, y0)

y0 += y1

g.DrawString(lblFifteen.Text, b, Brushes.Black, x2, y0)

g.DrawString(String.Format(fmtc, Taxes), b, Brushes.Black, x3, y0)

End Sub

End Class

### Credit Card Form

Public Class frmCC

Dim Month As Integer

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

If IsBlank(c) Then

MsgBox(c.AccessibleName & " is missing required info.", MsgBoxStyle.Information, "Error")

c.Focus()

Return True

End If

End If

If c.Name = "txtNumber" Then

If Not c.Text Like "####-####-####-####" Then

MsgBox(c.AccessibleName & " is not valid. Must be in ####-####-####-#### format.", MsgBoxStyle.Critical, "Error")

Return True

End If

ElseIf c.Name = "txtDate" Then

If Not c.Text Like "##/##" Then

MsgBox(c.AccessibleName & " is not valid. Must be in ##/## format.", MsgBoxStyle.Critical, "Error")

Return True

Else

Dim Index As Integer = c.Text.ToString.IndexOf("/")

Month = c.Text.ToString.Substring(0, Index)

If Month < 1 Or Month > 12 Then

MsgBox(c.AccessibleName & " is not valid. Must be between 01 and 12.", MsgBoxStyle.Critical, "Error")

Return True

End If

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnExit.Click

'Closes the window

If sender Is btnExit Then Close()

End Sub

Private Sub OK(sender As Object, e As EventArgs) Handles btnOk.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Determine if credit card information is invalid

Dim J, CheckNum As Integer

Dim NextChar As String

For J = 0 To txtNumber.Text.Length - 1

NextChar = txtNumber.Text.Substring(J, 1)

If NextChar Like "#" Then CheckNum = CheckNum + NextChar

Next

CheckNum = CheckNum.ToString.Substring(CheckNum.ToString.Length - 1, 1)

Select Case Month

Case 10 : Month = 1

Case 11 : Month = 2

Case 12 : Month = 3

End Select

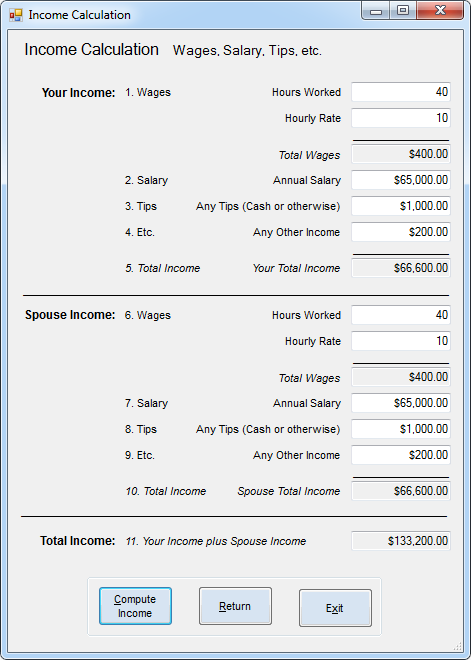
If CheckNum <> Month Then MsgBox("The month code of " & Month & " does not match the check digit of " & CheckNum & ".", MsgBoxStyle.Critical, "Error") Else Close()

End Sub

End Class

# Part C – Enhancements

## Income Calculation



Public Class frmIncome

Public Function IsInputInvalid() As Boolean

'Tests if any input is invalid

Dim Controls = From c In Me.Controls Order By c.TabIndex

For Each c In Controls

If c.AccessibleDescription = "Required" Then

Dim num = CZer(c.Text)

If IsNegative(num) Then

MsgBox(c.AccessibleName & " is not valid. Must be positive.", MsgBoxStyle.Critical, "Error")

c.Focus()

Return True

End If

End If

Next

Return False

End Function

Private Sub UtilityButtons(sender As Object, e As EventArgs) Handles btnReturn.Click, btnExit.Click

'Returns to the previous form

If sender Is btnReturn Then Close()

'Ends execution

If sender Is btnExit Then End

End Sub

Private Sub Calculate(sender As Object, e As EventArgs) Handles btnCompute.Click

'Runs invalid input test

If IsInputInvalid() Then Exit Sub

'Stores amounts in local variables and formats totals as currency for your income

Dim YHours = CZer(txtYHours.Text)

txtYHours.Text = YHours

Dim YRate = CZer(txtYRate.Text)

txtYRate.Text = YRate

Dim YOHours As Integer

If YHours > 40 Then YOHours = YHours - 40 Else YOHours = 0

Dim L1 As Decimal

If YOHours = 0 Then L1 = YHours \* YRate Else L1 = (40 \* YRate) + (YOHours \* (YRate \* 1.5))

txtYWages.Text = FormatCurrency(L1)

Dim L2 = CZer(txtYSalary.Text)

txtYSalary.Text = FormatCurrency(L2)

Dim L3 = CZer(txtYTips.Text)

txtYTips.Text = FormatCurrency(L3)

Dim L4 = CZer(txtYEtc.Text)

txtYEtc.Text = FormatCurrency(L4)

Dim L5 = L1 + L2 + L3 + L4

txtYIncome.Text = FormatCurrency(L5)

'Stores amounts in local variables and formats totals as currency for spouse income

Dim SHours = CZer(txtSHours.Text)

txtSHours.Text = SHours

Dim SRate = CZer(txtSRate.Text)

txtSRate.Text = SRate

Dim SOHours As Integer

If SHours > 40 Then SOHours = SHours - 40 Else SOHours = 0

Dim L6 As Decimal

If SOHours = 0 Then L6 = SHours \* SRate Else L6 = (40 \* SRate) + (SOHours \* (SRate \* 1.5))

txtSWages.Text = FormatCurrency(L6)

Dim L7 = CZer(txtSSalary.Text)

txtSSalary.Text = FormatCurrency(L7)

Dim L8 = CZer(txtSTips.Text)

txtSTips.Text = FormatCurrency(L8)

Dim L9 = CZer(txtSEtc.Text)

txtSEtc.Text = FormatCurrency(L9)

Dim L10 = L6 + L7 + L8 + L9

txtSIncome.Text = FormatCurrency(L10)

'Calculates the total or your and spouse income, saves in global variable, and then formats as currency

Income = L5 + L10

txtIncome.Text = FormatCurrency(Income)

End Sub

End Class