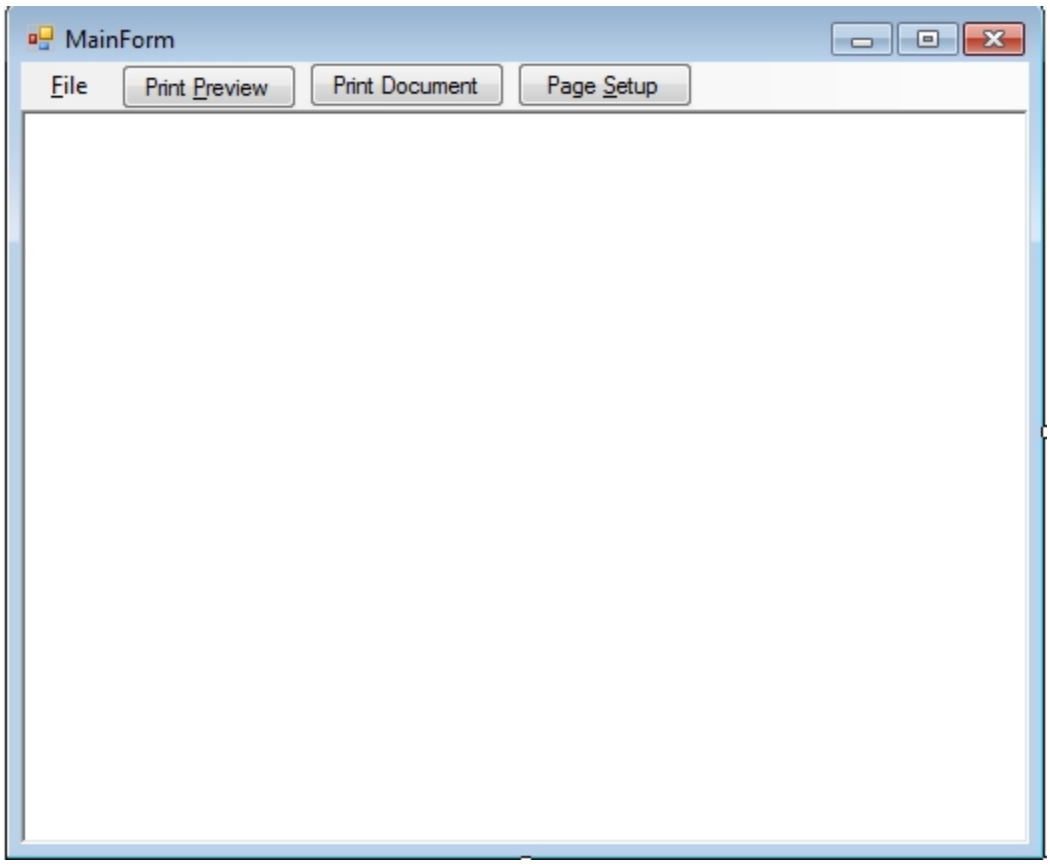
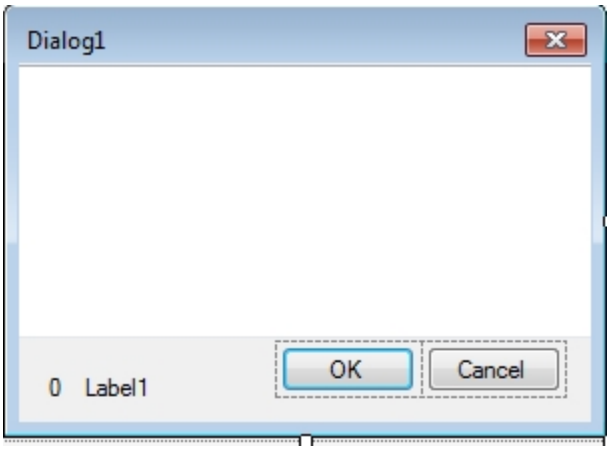
Print Document

' Requires a form named MainForm And a Dialog1 Box with a textbox
See Below: MainForm needs 3 buttons = btnPrintPreview, btnPrintDialog, btnPageSetup and a Richtextbox named txtDocument
' Place a PrintDialog1 control a PrintDocument1 Control a PrintPreviewDialog1 on the from with a RichTextBox1 that you are Printing From
' Dialog1 needs a Timer1. a textbox1, Lable1 and Lable2

```
    Private Sub PrintToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
PrintToolStripMenuItem.Click
        Dim MyDur As String
        MyDur = CurDir()
        If My.Computer.FileSystem.DirectoryExists(MyDur & "\Temp") Then
            logDirectoryProperties = My.Computer.FileSystem.GetDirectoryInfo(MyDur &
"\Temp")
        Else
            MkDir(MyDur & "\Temp")
        End If

        MyDur = CurDir() & "\Temp"
        If RichTextBox1.Text.Length >= 1 Then
            RichTextBox1.SaveFile(MyDur & "PrintTemp.rtf")
            SaveMyPath = MyDur & "PrintTemp.rtf"

            MainForm.Show()
            MainForm.txtDocument.LoadFile(SaveMyPath) ' = Me.RichTextBox1.Text
            Dialog1.Label1.Text = SaveMyPath
            Dialog1.TextBox1.Text = "Your document has been sent to the printer" & vbCrLf &
MyPath
        Else
            MsgBox("you must place text in the text box" & vbCrLf & "    OR first open a
file to  Print")

        End If
    End Sub

' Code for MainForm below
```






```
Imports System.Drawing.Printing
Public Class MainForm


    Dim MyPath As String
    ''' <summary>
    ''' The PrintDialog allows the user to select the printer that they want to print
    ''' to, as well as other printing options.
    ''' </summary>
    Private Sub btnPrintDialog_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPrintDialog.Click
        PrintDialog1.Document = PrintDocument1
        If PrintDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            PrintDocument1.Print()
            Dialog1.Show()
```

```
            Dialog1.Visible = True

            MsgBox("pit stop")

        End If
    End Sub

    '
    ''' <summary>
    ''' The PrintPreviewDialog is associated with the PrintDocument as the preview is
    ''' rendered, the PrintPage event is triggered. This event is passed a graphics
    ''' context where it "draws" the page.
    ''' </summary>
    Private Sub btnPrintPreview_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPrintPreview.Click
        Try
            PrintPreviewDialog1.Document = PrintDocument1
            PrintPreviewDialog1.ShowDialog()
        Catch exp As Exception
            MsgBox("An error occurred while trying to load the " & _
                "document for Print Preview. Make sure you currently have " & _
                "access to a printer. A printer must be connected and " & _
                "accessible for Print Preview to work.", MsgBoxStyle.OkOnly, _
                 Me.Text)
        End Try
    End Sub

    ''' <summary>
    ''' Page setup lets you specify things like the paper size, portrait,
    ''' landscape, etc.
    ''' </summary>
    Private Sub btnPageSetup_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPageSetup.Click
        With PageSetupDialog1
            .Document = PrintDocument1
            .PageSettings = PrintDocument1.DefaultPageSettings
        End With

        If PageSetupDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            PrintDocument1.DefaultPageSettings = PageSetupDialog1.PageSettings
        End If
    End Sub

    ''' <summary>
    ''' Handles the Form's Load event, initializing the TextBox with some text
    ''' for printing.
    ''' </summary>
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        'txtDocument.Text = My.Resources.Address
    End Sub


    ''' <summary>
    ''' PrintPage is the printing event. This event gets fired for every
    ''' page that will be printed. You could also handle the BeginPrint and EndPrint
    ''' events for more control.
    '''
    ''' The following is very
    ''' fast and useful for plain text as MeasureString calculates the text that
    ''' can be fitted on an entire page. This is not that useful, however, for
```

```
    ''' formatted text. In that case you would want to have word-level (vs page-level)
    ''' control, which is more complicated.
    ''' </summary>
    Private Sub pdoc_PrintPage(ByVal sender As Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
        ' Declare a variable to hold the position of the last printed char. Declare
        ' as static so that subsequent PrintPage events can reference it.
        Static intCurrentChar As Int32
        ' Initialize the font to be used for printing.
        Dim font As New Font("Microsoft Sans Serif", 12)

        Dim intPrintAreaHeight, intPrintAreaWidth, marginLeft, marginTop As Int32
        With PrintDocument1.DefaultPageSettings
            ' Initialize local variables that contain the bounds of the printing
            ' area rectangle.
            intPrintAreaHeight = .PaperSize.Height - .Margins.Top - .Margins.Bottom
            intPrintAreaWidth = .PaperSize.Width - .Margins.Left - .Margins.Right

            ' Initialize local variables to hold margin values that will serve
            ' as the X and Y coordinates for the upper left corner of the printing
            ' area rectangle.
            marginLeft = .Margins.Left ' X coordinate
            marginTop = .Margins.Top ' Y coordinate
        End With

        ' If the user selected Landscape mode, swap the printing area height
        ' and width.
        If PrintDocument1.DefaultPageSettings.Landscape Then
            Dim intTemp As Int32
            intTemp = intPrintAreaHeight
            intPrintAreaHeight = intPrintAreaWidth
            intPrintAreaWidth = intTemp
        End If

        ' Calculate the total number of lines in the document based on the height of
        ' the printing area and the height of the font.
        Dim intLineCount As Int32 = CInt(intPrintAreaHeight / font.Height)
        ' Initialize the rectangle structure that defines the printing area.
        Dim rectPrintingArea As New RectangleF(marginLeft, marginTop, intPrintAreaWidth,
intPrintAreaHeight)

        ' Instantiate the StringFormat class, which encapsulates text layout
        ' information (such as alignment and line spacing), display manipulations
        ' (such as ellipsis insertion and national digit substitution) and OpenType
        ' features. Use of StringFormat causes MeasureString and DrawString to use
        ' only an integer number of lines when printing each page, ignoring partial
        ' lines that would otherwise likely be printed if the number of lines per
        ' page do not divide up cleanly for each page (which is usually the case).
        ' See further discussion in the SDK documentation about StringFormatFlags.
        Dim fmt As New StringFormat(StringFormatFlags.LineLimit)
        ' Call MeasureString to determine the number of characters that will fit in
        ' the printing area rectangle. The CharFitted Int32 is passed ByRef and used
        ' later when calculating intCurrentChar and thus HasMorePages. LinesFilled
        ' is not needed for this sample but must be passed when passing CharsFitted.
        ' Mid is used to pass the segment of remaining text left off from the
        ' previous page of printing (recall that intCurrentChar was declared as
        ' static.
        Dim intLinesFilled, intCharsFitted As Int32
        e.Graphics.MeasureString(Mid(txtDocument.Text, intCurrentChar + 1), font, _
                    New SizeF(intPrintAreaWidth, intPrintAreaHeight), fmt, _
                    intCharsFitted, intLinesFilled)
```

```
        ' Print the text to the page.
        e.Graphics.DrawString(Mid(txtDocument.Text, intCurrentChar + 1), font, _
            Brushes.Black, rectPrintingArea, fmt)

        ' Advance the current char to the last char printed on this page. As
        ' intCurrentChar is a static variable, its value can be used for the next
        ' page to be printed. It is advanced by 1 and passed to Mid() to print the
        ' next page (see above in MeasureString()).
        intCurrentChar += intCharsFitted

        ' HasMorePages tells the printing module whether another PrintPage event
        ' should be fired.
        If intCurrentChar < txtDocument.Text.Length Then
            e.HasMorePages = True
        Else
            e.HasMorePages = False
            ' You must explicitly reset intCurrentChar as it is static.
            intCurrentChar = 0
        End If
    End Sub

    Private Sub exitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles exitToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub MainForm_Resize(sender As Object, e As EventArgs) Handles Me.Resize
        txtDocument.Width = Me.Width - 16
        txtDocument.Height = Me.Height - 70
        txtDocument.Top = 66
    End Sub

End Class
```