50 points DUE: Specified on Canvas

**Assignment Purpose**

This assignment will give you practice utilizing standard Font dialog, RichTextBox control, menus, building, and using a DLL in your application. You will need to work without a partner for this assignment.

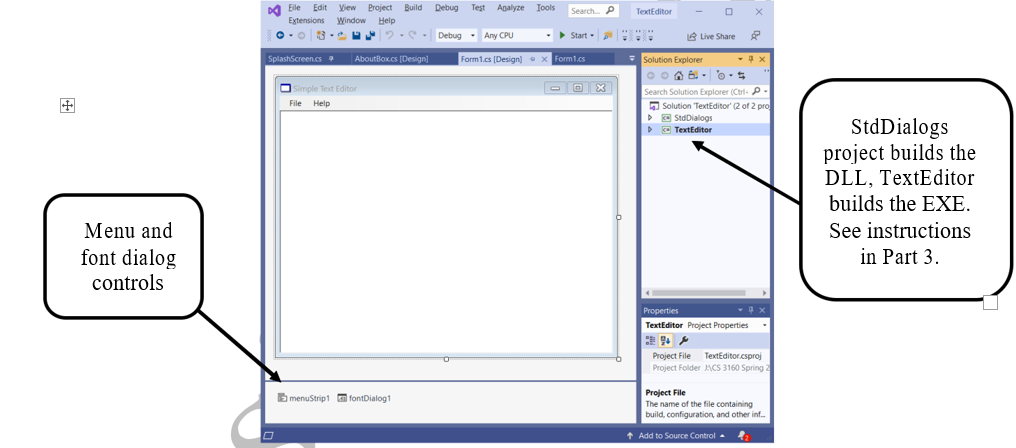
**This application is required as part of the take home programming portion of the FINAL EXAM for this class. It will be worth 50 points.**

**Mandatory Instructions**

*Part 1: Create a simple text editor application*

~~Create an application whose form has the main menu, a rich text box, and calls up a standard Font Dialog. The default font for the form and the rich text box should be 10-point Arial.~~

~~The File menu contains three commands: Clear, Font… , and Exit. The Help menu has just one command: About… The rich text box has the Dock property set to Fill.~~



* ~~The ShowEffects property of the font dialog object should be set to False~~.
* ~~Implement the commands on the File menu for this simple application~~.
* ~~The Clear command should remove all text from the rich text box.~~
* ~~The Font command should display the font dialog and use the resulting font to change the font for all of the text appearing in the rich text box.~~
* ~~The selected font should become the default font for the rich text box~~.
* ~~The Exit command should terminate the application. Run the program to verify it's working before continuing.~~

50 points DUE: Specified on Canvas

*Part 2: Change the icon for this application*

1. ~~Create one of your own. Save the icon in the project folder.~~
2. ~~Use "Add existing item" to add the icon to the project.~~
3. ~~Go to the project's properties and change the icon for the project to the one you selected. Just type in the name of the ICO file. Don't browse for it.~~
4. ~~Compile and run. Is the icon for the EXE file the correct one?~~
5. ~~Change the form icon to match.~~

*Part 3: Add a DLL project to the Solution*

1. ~~Right-click on the Solution and add a new project to the Solution called StdDialogs. This Solution should be a "Class Library" and NOT a Windows application.~~
2. ~~Delete the file called Class1.cs in the resulting project~~.
3. ~~Add a new form to this project called "AboutBox.cs". This form will be used to display an About box for a variety of applications. The form should not be resizable.~~
4. ~~The form should contain a picture box, three labels (one below the other, each the width of the window, with center justification for the text), and an OK button. The font size for the first label should be larger than the size of the other two. The Text property of the labels can be anything at all. Give a good name to each control on the form. The OK button merely closes the form.~~
5. ~~The constructor of this form should accept an Image and three strings~~. ~~The image will be associated with the picture box. The size of the picture box should be changed to match the size of the image passed in~~. ~~The strings should become the text of the labels on the form.~~ **~~Do not~~** ~~hard-code. Both must be set dynamically.~~
6. ~~Build your DLL project, but don't try to execute~~.

*Part 4: Add a logo image file to the application*

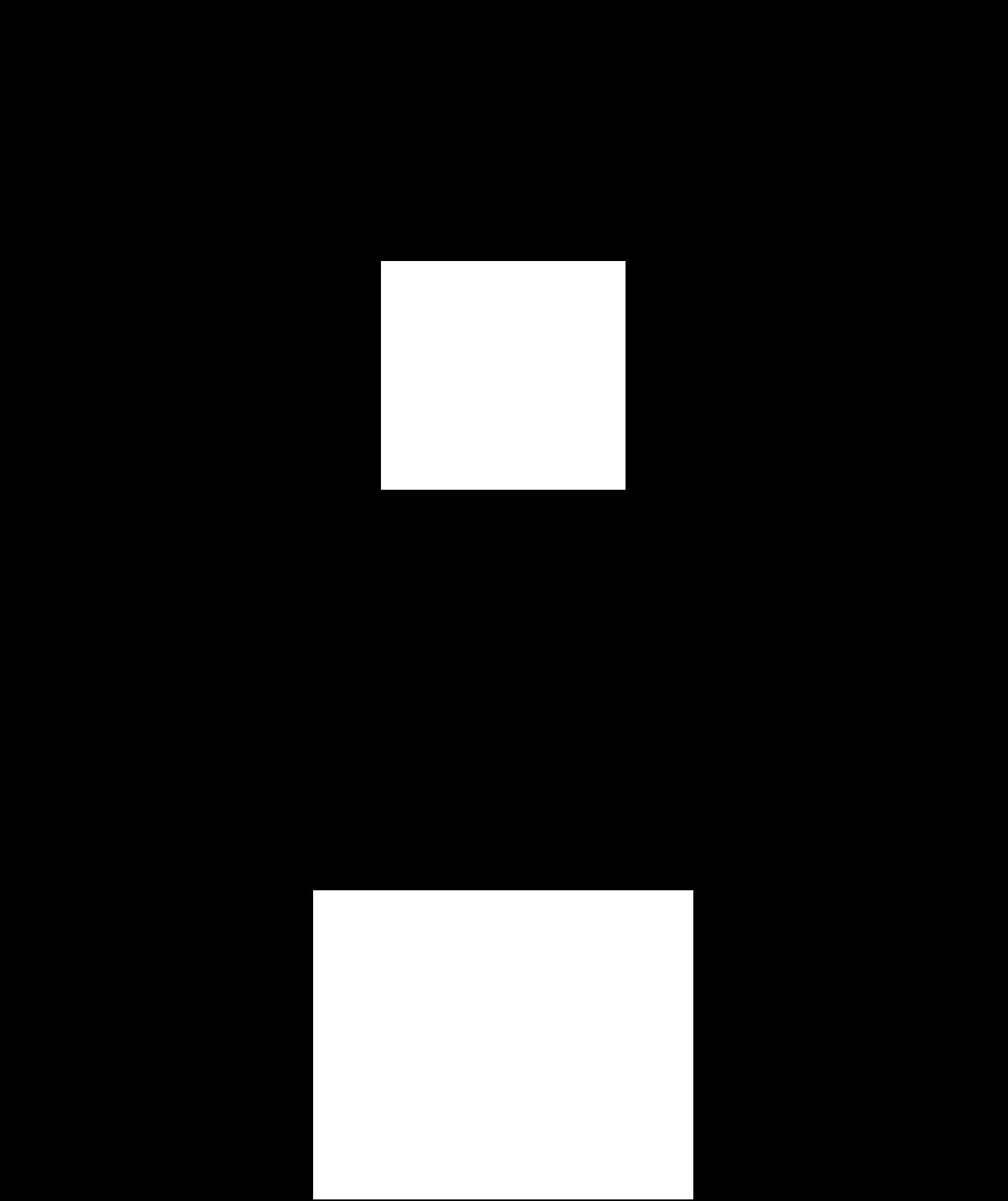
1. ~~Choose a small image file to display in the about box and save it in the application folder.~~
2. ~~Add this image to the project and change the Build Action property to "Embedded Resource".~~
3. ~~Add private class-level data to the form: private Image mAppLogo;~~
4. ~~Add code to the Form1 constructor to load the logo from your file into the Image object. The following code should work (if your main project is called SmithApp7 and your image file is logo.jpg):~~

System.Reflection.Assembly myAssembly = System.Reflection.Assembly.GetExecutingAssembly(); System.IO.Stream myStream = myAssembly.GetManifestResourceStream("SmithApp7.logo.jpg"); appLogo = Image.FromStream(myStream);

~~NOTE: If you have a hyphen in the project name, it will be replaced with the underscore character in the call to GetManifestResourceStream.~~

*Part 5: Add the About box to the application*

1. ~~In the main application, add a reference to the DLL. (Remember how? Right-click on References in Solution Explorer, and then…) You'll also want to add a "using" statement for the namespace.~~
2. ~~For the About command, display the About box found in the DLL. Pass in the logo image and three strings (maybe program name, copyright date, programmer name). Run the program and verify that everything is working~~.



*Part 6: Add a splash screen to the DLL and use it in the application*

A splash screen is visible for a few seconds when you first start an application. The format will be very similar to the about box, with these changes:

* ~~There is no OK button.~~
* ~~The form will have the following properties: TopMost=True, FormBorderStyle=None, StartPosition= CenterParent.~~
* ~~The form will have a Timer object.~~
* ~~The constructor will pass in the number of milliseconds that the splash screen should be visible, in addition to an image and three strings. Use 5000 ms.~~
* ~~The constructor initializes the Interval property of the timer to the value passed in as the parameter.~~
* ~~When the Tick event occurs, the form is closed.~~

~~When you have added the splash screen to the DLL, display the splash screen from the Load event of Form1.~~

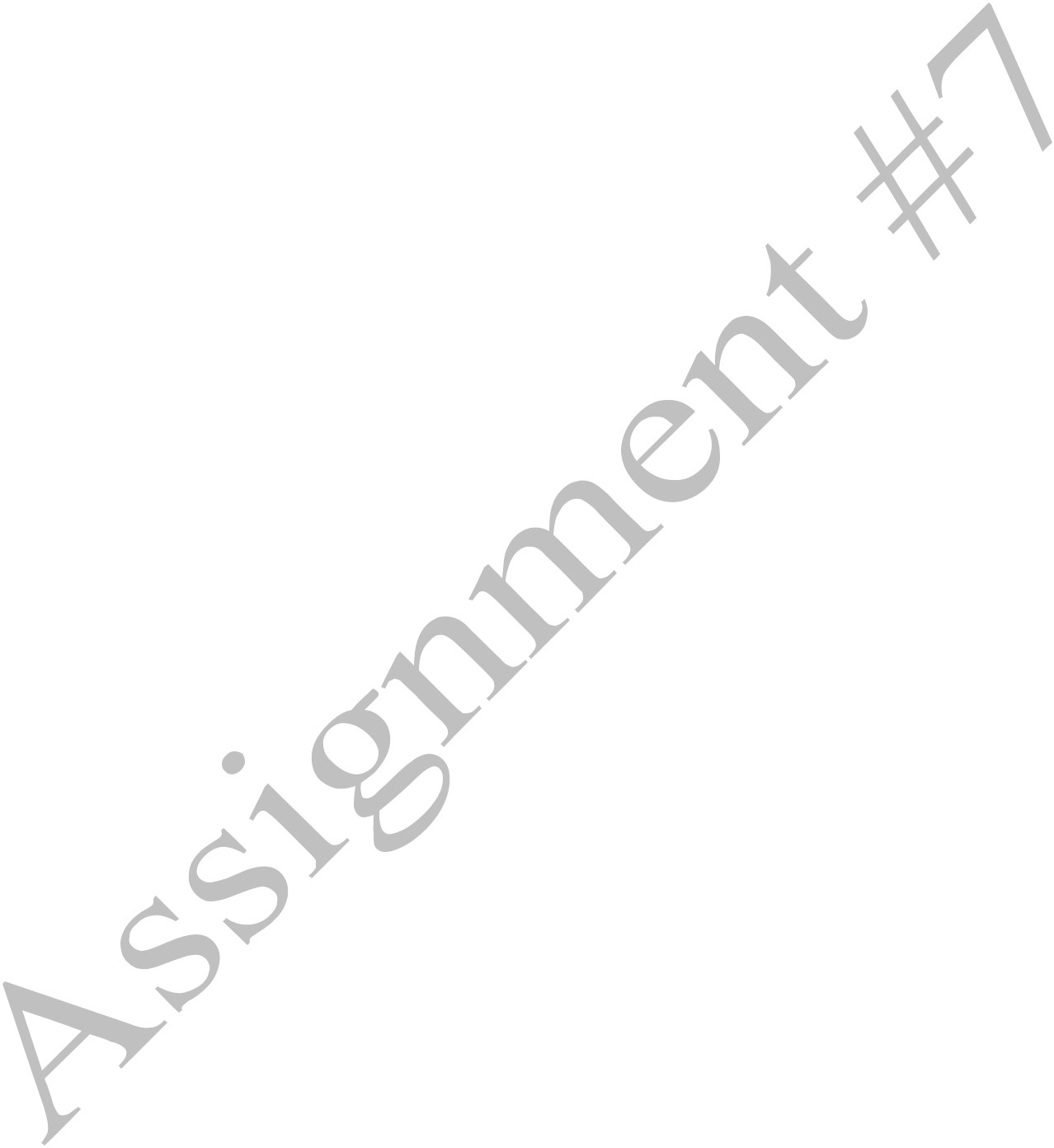
Run your application and make sure everything is working properly.

CS3160, Instructor: Carlson

50 points DUE: Specified on Canvas

**Code Documentation**

Each source file should contain a documentation header with a description, author name, class information (CS3160, Spring/Fall 20xx). Each function should have a documentation header with a minimally function name, description, parameters, return value. Use proper program style at all times, which means indentation, alignment, and whitespace. Utilize self-documenting code which means use mnemonic, i.e., meaningful variable/function/namespace/class names, etc.



**What to turn in?**

Submit a compressed (.zip) solution folder via Canvas.

Make sure that all necessary files to compile/link/execute your projects are provided in your solution folder.

Here are some files that may be required…

1. All required C# source files (.cs)
2. makefile (if the Solution requires it)
3. The entire Visual Studio 2019 solution folder (if the Solution requires it)

CS3160, Instructor: Carlson