# Unit Test #3

1. Write the architecture and pseudo-code of the attached Presidents.exe Windows Forms application. Run the application as Administrator, and be sure to hover over everything, type in every text box, and answer every question correctly!

If the Wikipedia "Page Preview" feature is enabled, disable it by hovering over a link, selecting the gear icon, and unchecking the "Page previews" checkbox. Refer to lecture 13-1 for step-by-step instructions if you need them.

**List all controls.** Note that depending on how you implement the code there could be ~47 or ~62 controls (including the Form control). You could either import each image into a separate PictureBox, using the Image property, or you could have one PictureBox control and set the ImageLocation property at run-time to point to the image file at a hard drive location (such as c:/temp/GeorgeWashington.jpeg) or my Banjo site. I uploaded all of the images to <http://people.rit.edu/dxsigm> (for example <https://people.rit.edu/dxsigm/GeorgeWashington.jpeg>)

**Include the pseudocode for all constructor code and all event handlers.**

1. Based on #1, re-create the Presidents.exe application. Be sure to refer to "Windows Forms Controls" in myCourses along with the associated projects we have worked on in this unit. Ensure that you use the mobile version of wikipedia (prefix your URL's with <https://en.m.wikipedia.org/wiki/>)
2. User Interface From Hell

You’ve probably seen enough user interfaces to have seen some that aren’t great. Your homework is to create a user interface that will make the user’s life as miserable as possible.

You should try to be as creative as possible: Things like moving buttons, text fields that always drop the letter “e” from a word, random font changes, GUIs that look different each time you start them or horrible color schemes are a good start (but come on, that’s easy. Do your worst.)

# Requirements:

Your GUI will need to include at least **one thread** and **five different** *types* of controls.

The thread must interact with one of the controls to change the appearance or content on a form.

Five buttons do not count as five controls – they are all the same type.

At least one of each type of control needs to **be active**. In other words, they must respond to user input in some way (don’t just make a bunch of “inactive” controls or labels). Examples include:

* Button presses
* Mouse events, such as a mouse entering a button or text area.
* Keyboard events
* List selections
* Any other recognized events

Your events should be **varied in their outcome**. Don’t just use Message Boxes for everything. Change colors. Change or delete text. Move elements. Perform incorrect calculations. Format numbers or text strangely. Have things happen on timers.

Your GUI must have at least **3 different windows** that communicate with each other in some way. (Change text or other properties, etc.) The main form counts as one, so you’ll need at least 2 more.

**Your project must have a theme.** It is not sufficient to have some random collection of components; the GUI must actually have a purpose. It could be data collection for an order, filling in a form, creating bad haiku poetry, or anything else you can come up with. Use your imagination. It doesn’t have to necessarily “work”, but it should seem like it would work if it wasn’t so horrible to use.

Do not create a GUI that runs amok and deletes files or otherwise screws things up. There has to be a way to locate the main window and kill it off. (**This is serious**, don’t do anything malicious).

Be as **creative** as possible. You could look at paper forms, web data entry systems, horrible game interfaces, SIS, etc. for inspiration.

Crashing is not considered part of a bad interface… You must perform basic error checking and implement try/catch blocks anywhere your code would possibly crash. This should prevent user errors and keep the entire form from crashing. (Possible opportunities for a try/catch: Inputting numbers vs. text, parsing to specific data types, division by zero, etc.)

# Documentation

Your documentation must include all of the following:

* The theme of your form and why you chose that theme
* A detailed description of what the form does
* A description of each event and what occurs when the event is triggered
* How to stop the form (if necessary)

# The serious stuff

Do not include any infinite loops, or loops that run a ridiculous amount of times. (Let’s put a hard limit of 10 iterations on any particular event. This means a maximum number of MessageBoxes that can appear at one time is 10). There has to be a way to locate the main window and kill it off. (This is serious, don’t do anything malicious).

Finally, make sure you understand the difference between “annoying” and “offensive” when it comes to an interface. Annoying is expected, offensive will be penalized. Something like a RickRoll is fine (although it’s been done before). Linking to or otherwise showing something offensive/gross/NSFW/etc. is not ok.

# Finally…

Have fun with it!

## Submission

Upload a Word Document for #1 and the documentation for #3, and the GitHub URL's #2 and #3 to the corresponding myCourses dropbox.