**Lab 0.1**
**Git and Familiarity with Command Line**

This is the second half of the lab. Congrats on cloning a Git repository using the command line! There are a couple more steps to practice editing a file and going through the process of commiting to *git*.

1.) Make sure that your working directory is *inside* of the directory that you cloned from *github*, you can use the command `pwd` to check your current working directory in the terminal.

2.) Use the text editor *gedit* to open a new, empty text file with the name `readme.txt`. *gedit* is a simple and easy to use text editor that is accessible from the command line, and is good for making quick changed to a file. You can open a file with it from the command line by typing `gedit <filename>`, where you replace `<filename>` with the name of the file that you want to open (in this case, `readme.txt`).

3.) Write something, *anything* in the file, and **save** it. Make sure you write at least one line of text. Speak your mind, say hello to the world, or to git, or just scribble. It situations like this, I tend to write `dfasdfalskdjgh` (or something similar). Make sure you **save** the file, then quit *gedit*.

4.) Type `cat readme.txt` in your terminal. What do you think the command `cat` does?

5.) Ok – now it is time to stage, commit, and push the changes you made in the *local* git repository to the git repository on *github*. There are three steps to this. First, you must choose from which files you want to save changes. In the case of this simple lab, we want to include changes in *all* files, including the new one. Type `git add .` to *stage* all files in your current directory for commit.

6.) The files are staged, now they have to actually be commited! With *git*, every commit has a message associated with it. The most simple way to commit and include a message is the command `git commit –m "{{ my message }}"` where `{{ my message }}` is the message that you want to associate with that commit. *Note: exclamation marks and other special characters may cause this command to act unexpectedly.*

7.) Now, we have to *push* the commit that you made back to *github*. You can do this simply by typing the command `git push`. After you type that (and press enter to run it), you will see some text describing what is happening. Once the command completes, go to your *github* account and check out how your repository now includes the changes that you made to it.

*Note: at any point, you can type* `git push` *to literally check the status of your git repository. This will be much more useful as the things that you know how to do with git, and the way that you use it becomes more complex.*