Office Hours ++ 3
Using and Customizing Vim

Due: Monday, April 2nd, 11:59PM (Hard Deadline)

Submission Instructions

Submit a link to your `.vimrc` to this link: https://goo.gl/forms/bAgdq0v5mKB6uAEN2. This could be from inside the dotfiles repository you created in Homework 12, or from some other online storage site of your choice. Ensure that you make it publicly viewable to anyone who has a link.

Please add comments in your file to indicate where the feature you built is, and what section it is from, so that we are able to find it easily.

1 Building a `.vimrc`

A `.vimrc` is a configuration file for Vim that carries settings that are loaded by the editor on startup. These configurations can range from theme personalisations and key mappings, to custom functions and integrations. As we have seen throughout this course, since it is essentially just a text file, the `vimrc` can be edited in order to customize Vim. Spending time to build up a `vimrc` that suits your preferences and work styles best can save you a lot of time in the future, and as you continue to use Vim it only continues to grow. By the end of it all, a good `vimrc` should make it so that using your editor is second nature to you and requires no deeper thought, which will allow you to spend more time and energy on actually writing high-quality code.

Your `vimrc` is located in your `$HOME` directory by default, but you have the flexibility to move it to a different directory and create a symbolic link in your `$HOME` to your `vimrc` in this other directory (as you would have done if you completed Homework 12).

The Assignment

While having a good `vimrc` is extremely helpful, it is often difficult to start building one at the start. For this assignment, your task is to begin putting together a `vimrc` that suits your needs best. To receive credit, you will need to pick and implement **at least ONE customization FROM EACH of the following sections**. In the end, this means that you will have added **at least THREE customizations** to your `vimrc`.

1.1 Package Managing

Choose **at least one** of the following customizations to implement. Make sure to add comments in your `vimrc` to indicate the feature you chose to write and which section it come from.

The customizations in this section involve making use of external plugins or packages in Vim, and the best way to do, as you would remember from last week's lecture, by using a package manager. There are a lot of good package managers available for Vim, such as Vundle and Pathogen, so choose one that looks best to you and get it set up.

For all of the following customizations, use your package manager to install the plugin and then configure the plugins usage options in your vimrc.

- ☐ Add a snippets package and assign custom key bindings for selecting snippets
- ☐ Add a code completion engine (eg. YouCompleteMe) and assign custom key bindings to use for code completion
- ☐ Add the ability to navigate directories and open files through Vim
- ☐ Add some other package that you think will be useful and configure it to work the way that you think it should! If you choose this option, in your comments include why you chose to use this package.

1.2 Appearance

Choose **at least one** of the following customizations to implement. Make sure to add comments in your `vimrc` to indicate the feature you chose to write and which section it come from.

- ☐ Add a custom theme which changes the color schemes or layout of the editor
- ☐ Display warnings or source control markings next to line numbers
- ☐ Customize the bottom ribbon of Vim to show additional information, such as the number of words, the language being written in, and a summary of any linter warnings.
- ☐ Change the appearance in some other meaningful way! If you choose this option, explain in your comments what your change does and why you chose to do this.

1.3 Vimscripting and Other Functionalities

Choose **at least one** of the following customizations to implement. Make sure to add comments in your `vimrc` to indicate the feature you chose to write and which section it come from.

- ☐ Write a Vimscript that removes unused `import` or `include` from a file
- ☐ Display warnings or source control markings next to line numbers
- ☐ Use abbreviations as shortcuts for common strings you type repeatedly in your code. Some suggestions include:

```
#include <iostream>
using namespace std;
return 0;
cout <<
endl;
RME's (if you're in 280)
```

- ☐ Create 2 mappings. One that splits the current editor and open up your `.vimrc` file for editing, and then another that saves your changes and closes the split
- ☐ Add some other useful function to Vim. As before, if you choose this option, include a description of what you chose to build and why you chose to do so in your comments.