Basic 6
Text Editors

EECS 201 Fall 2020

Submission Instructions

Answer the bolded text that begins with a "**Q**". This assignment is an "online assignment" on Gradescope, where you fill out your answers directly.

Optional Readings

Previous iterations of this class included some optional readings with this week. You may find the ideas and topics they bring up interesting.

Interrupting Developers and Makers vs Managers

http://thetomorrowlab.com/2015/01/why-developers-hate-being-interrupted/
Why developers hate being interrupted, by Derek Johnson at The Tomorrow Lab.
http://www.paulgraham.com/makersschedule.html
Maker's Schedule, Manager's Schedule, by Paul Graham, co-founder of Y Combinator.

Software Engineering in the Real World

http://blogs.msdn.com/b/peterhal/archive/2006/01/04/509302.aspx
What Do Programmers Really Do Anyway?, by Peter Hallam, then-Microsoft Developer

> *Why is 5 times more time spent modifying code than writing new code? The answer is that new code becomes old code almost instantly. Write some new code. Go for coffee. All of sudden you've got old code.*

http://www.joelonsoftware.com/articles/fog0000000069.html
Things You Should Never Do, Part I, by Joel Spolsky

> *It's harder to read code than to write it.*

When was the last time you saw a hunt-and-peck pianist? -Jeff Atwood, co-founder of Stack Overflow and Discourse. Some extras on the value of being a good typist and ditching the mouse for the keyboard.

1 Trying out fancy features

For this question, we'll be looking at the two big terminal editors: Vim and Emacs and exploring their capabilities. You may choose either of them to work with for this question. **You many only use Vim or Emacs in this part of the assignments.**
The intent of this question is to expose you to the more advanced features that other editors like Microsoft Notepad or GNU Nano may not have as well as building competency in terminal text editors. I encourage you play around with the editors and look into the neat features that they have.

If you're using Emacs, run it in non-GUI mode i.e. `emacs -nw` for this question.

To start, grab a copy of this file that's full of code:

```
# Example from http://web.mst.edu/~price/cs53/code_example.html
wget http://web.mst.edu/~price/cs53/KatsBadcode.cpp
```

---

1. Sometimes you will come across some ugly code. Your editor can help make it better. Open `KatsBadcode.cpp`. Among other issues, the really bad tabbing makes this hard to read.
   Q: Describe how to automatically fix the indentation for the whole file.

   Now imagine if you are editing code and you determine you can remove an if block, for example removing the `if` on line 28 of `KatsBadcode` and running its body unconditionally. You need to fix the indentation level of all 20 lines of the body.
   Q: Describe how to change the indentation level of a block of code.

2. Editing one file is nice, but often it's really useful to compare files side-by-side (source and headers, spec and implementation).
   Q: Describe the command sequence to create a window split (within your text editor) side-by-side with your current window, switch to it, and then open a file in that window.

   Try playing around with these two views, copy/paste code between them, bind them so they scroll together, resize them, add more splits.

3. Editors also have pretty nice integration with compilation tools. With `KatsBadcode.cpp` open, try typing `:make KatsBadcode` in Vim or `M-x compile<enter>make KatsBadcode` in Emacs. [1]

   First cool thing that happened: Yes, you can run `make` *without* any Makefile anywhere. We'll cover how that happened during build-system week.

   Building `KatsBadcode.cpp` will fail with several errors.
   Q: Describe how to navigate between compilation errors automatically. Hint: Vim has the "quickfix" feature and Emacs has "Compilation Mode"

4. While C-style languages support `/* block comments */`, others such as Python have no block comments and require you to put a `#` at the beginning of every line.
   Q: Describe how to efficiently comment out a large block of Python code. No, docstrings (`"""`) are NOT comments.

[1]Descriptions of `Emacs` commands are usually written as `C-c` or `M-x`, which mean "Ctrl+C" or "Meta+x" respectively. Meta is often mapped to the escape and/or alt key, `M-x` means press Escape and then x or hold alt and press x.

5. Many times you have a repetitive task that you need to do say 10 or 20 times. It's too short to justify writing a script or anything fancy, but it's annoying to type the same thing over and over again. As example, reformatting the staff list to a nice JSON object:

<pre>Boole,George
Lovelace,Ada
von Neumann,John
Hopper,Grace
Turing,Alan
Shannon,Claude
Dijkstra,Edsger
Hamilton,Margaret
Knuth,Donald</pre>
	<pre>staff = [
	    {"first": "George", "last": "Boole"},
	    {"first": "Ada", "last": "Lovelace"},
	    {"first": "John", "last": "von Neumann"},
	    {"first": "Grace", "last": "Hopper"},
	    {"first": "Alan", "last": "Turing"},
	    {"first": "Claude", "last": "Shannon"},
	    {"first": "Edsger", "last": "Dijkstra"},
	    {"first": "Margaret", "last": "Hamilton"},
	    {"first": "Donald", "last": "Knuth"},
	]</pre>

Editors support the record and replay of *macros*. With macros, you can start recording, puzzle out how to turn one line into the other, and then simply replay it for the rest. As a general rule, do not bother trying to be optimal or efficient, just find anything that works. Try starting with the column on the left and devising a macro to turn it into the column on the right.
Q: Describe how to record and replay a macro. *(you do not need to include the contents of your macro; make your best attempt at trying to make a macro that works, but don't worry if you are unable to get it to work perfectly)*

6. Editors understand balanced ()'s and {}'s.
Q: Describe how to jump from one token to its match, such as from the opening { on line 29 of `KatsBadcode` to its partner } on line 58.

7. Sometimes it's useful to run quick little commands. For example, your code needs to read from a file, but you can't remember if it's called `sample_data.txt` or `sampleData.txt`.
Q: Describe how to run 'ls' directly from within your editor.

Personalising your Editor (Ungraded but recommended)

Similar to how `~/.bashrc` configures `bash`, a `~/.vimrc` or `~/.emacs` file[2] can configure how your editor behaves. **Add something useful not listed above to your editor's configuration file.**

Q: Roughly how long did you spend on this assignment?

[2] These files do not exist by default, you have to create them.