Homework 4
Text Editors

EECS 201 Winter 2020

Submission Instructions

Answer the bolded text that begins with a bolded number e.g. "**1**", "**2**". Submit this assignment on Gradescope. **When submitting please select the right pages for each question or we will not grade the question**. You may find the free online tool PDFescape helpful to edit and fill out this PDF. You may also print, handwrite, and scan this assignment.

If you feel enterprising, you can edit the TeX file directly, compile it to a PDF, and submit that! I ask that if you do so, please make the answer text stand out from the rest of the homework.

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.
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`, remember to 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.
   1: 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.
   2: 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).
   3: 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.
   Optional: Describe how to navigate between compilation errors automatically.

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.
   4: Describe how to efficiently comment out a large block of Python code.

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:

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

   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.
   5: 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)*

[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.

6. Some final little things

   (a) Editors understand balanced ()'s and {}'s. **6: 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.**

   (b) 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`.
   7: 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.**

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

2 Remote work

One thing that can make life very convenient is learning how to work on remote machines. CAEN has a large number of machines available for students to use remotely, at `login.engin.umich.edu`. Let's check that out.
`ssh` is the **s**ecure **sh**ell program, and is used to securely log in to remote machines.
Run `ssh <your uniqname>@login.engin.umich.edu` to log into a CAEN machine. Take a look around, this is the same login environment and home directory as if you had sat down at a physical CAEN machine.
Use `vim` or `emacs` to create a file called `test.txt` on CAEN. Put some text in it, save, and quit.
Now try to run `gedit test.txt`.
1: What error do you get? Why?

Close your ssh session (log out of the terminal, either by typing `exit` or pressing `Ctrl-d` at an empty prompt). Then log in again with the additional `-X` flag to ssh: `ssh -X <your uniqname>@login.engin.umich.edu`.
Now try running `gedit test.txt` – great! Only, now you cannot do anything else in that terminal.
2: How can you keep using this terminal without killing the gedit session that is already running?
3: How should you launch `gedit` so that you can still use the terminal?
(Hint: remember we talked about shell job control a few weeks ago)

Try running a more graphically intensive program, such as the `eclipse` IDE or `matlab`. How does it compare to sitting at a physical CAEN machine? How does it compare to using VNC to log into CAEN? (And how good is your Internet connection right now ;) ?)

You may also find the **s**ecure **c**o**p**y program, `scp`, and `rsync` useful for moving files between your machine and the remote machine. No question on either of these, just letting you know they exist.

4: Roughly how long did you spend on this assignment? ____________________