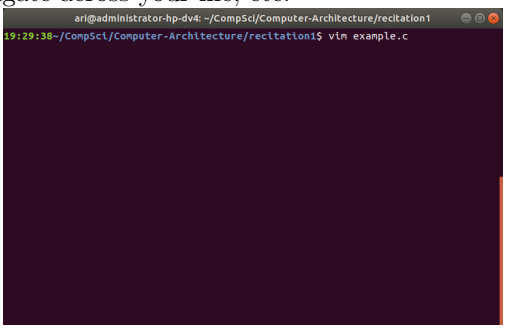
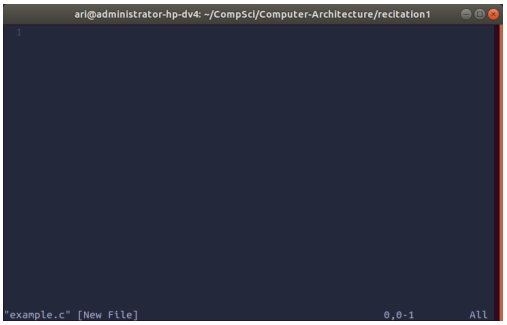
Vim — Example Usage

Ari Feiglin (ari.feiglin.biu@gmail.com)

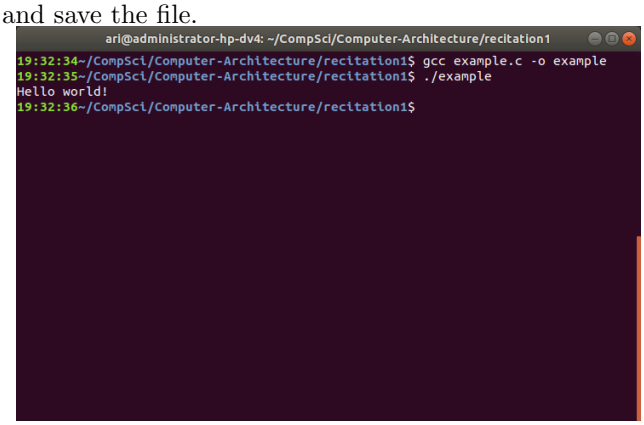
1. Create a new blank file with vim. You are now in *normal mode*, where you can access other modes, navigate across your file, etc.





2. Go into *insert mode* (by hitting the `i` key), and type text as you would normally in a text editor.

3. Go back into normal mode (`ESC`) and then into command mode (colon, `:`). Save the file with `w` (be sure to hit enter afterward). This will put you back into normal mode, go back into command mode and quit (`q`). These two commands can also be combined into one: `:wq`. Compile and run the C file to show that vim did indeed create and save the file.



4. Use vim to open the provided example file (foo.c).

5. Move the cursor utilizing the `h`,`j`,`k`,`l` keys.

6. Go to the bottom `G` and top `gg` of the file.

7. Go to a specific line, eg. `69G`.

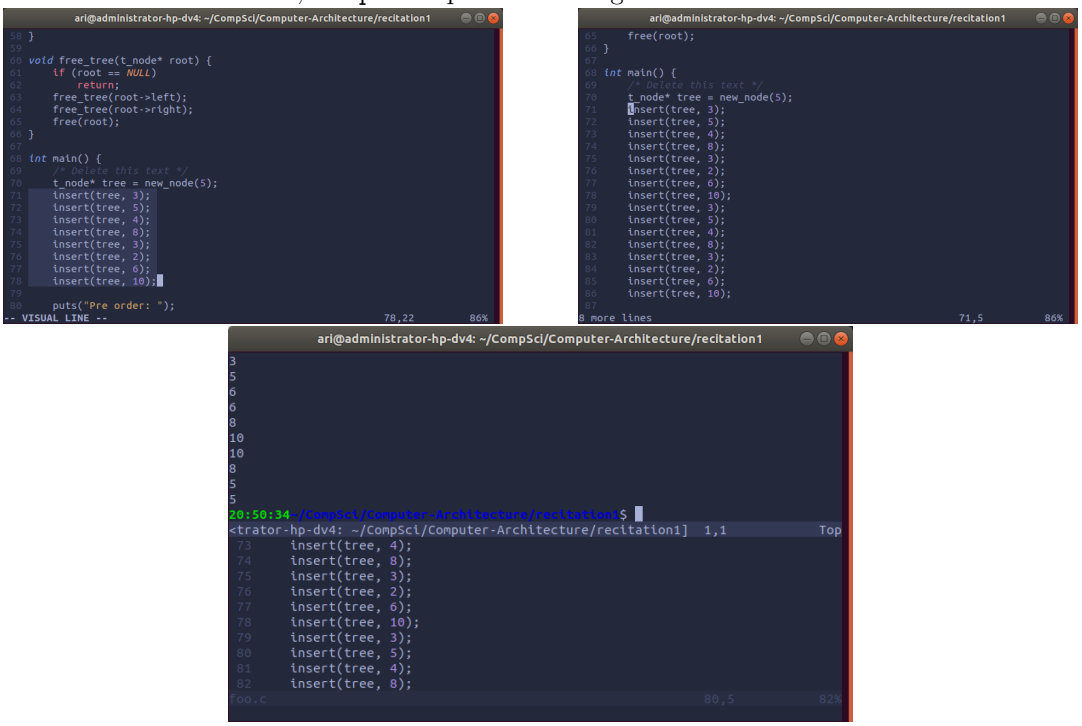
8. Navigate to line 69, and delete a character using `x`, a word with `dw`, and the entire line with `dd`.

9. Demonstrate that the terminal can accessed from within vim by `:term` and navigate to the terminal (to navigate between windows, first do `CTRL-w`, then a navigation key, `h,j,k,l`. In this case it will be `CTRL-w k`). Compile and run the program. (Remember to save any changes with `w` before running!) Navigate back to the editor with `CTRL-w j`.

10. Line 86 has a typo — it should be "Good bye!". Navigate to line 86, to the correct column and fix this issue (navigate on top of the !, hit `i`, and insert an e).

11. Wherever you feel like it, show that you can go to the end of a line with `$`, the beginning with `0`, and to the first non-whitespace character with `^`.

12. Navigate to line `70` and go into *visual mode* (linewise: `V`), select until line `78` and yank the contents using `y`. Then paste the contents above the cursor (the cursor will be brought back to line `70`) using `P`. To paste the contents below the cursor, use `p`. Compile and run again.



13. Search for the pre-order function using the search command `/` (ie. do something like `/pre_order`). Navigate through the results with `n` (next) and `N` (previous).

14. The t_ in t_node is redundant: change all occurrences of t_node with node by `:%s/t_node/node/g`. Here `s` is short for substitute and has the following usage: `[range]s/pattern/string/[flags] [count]`. `%` is a range which is short for `1,$` meaning from the first to last line. In general ranges have the form `beginning,end`. `g` is a flag which replaces every occurrence on each line being replaced, without it only the first occurrence is replaced.

15. Never mind, the t_ is actually useful: undo the previous change with `u`. But you can always redo a change with the command `:redo`.

16. Exit vim. Then enter vimtutor, and navigate around a bit. You don't need to follow the instructions but make sure the students are aware of its existence.

Important notes:

1. It is not necessary to follow these instructions verbatim, these are only some guidelines I think may be useful. Alternatively you could just follow vimtutor entirely.
2. To exit multiple vim windows at once (for example an editor and a terminal window), do `:qa`. To also save, do `:wqa`. `:q` closes only the current selected window, you must add the `a` to close all windows.
3. To scroll within vim's terminal, do `CTRL-w N`. This brings you to "Terminal Normal mode", where you can navigate around the terminal similar to as you would in normal mode.
4. If you see that vim says something like "recording @q" at the bottom — you have accidentally started recording a key. Just hit `q` in normal mode to stop this. It's not significant but it can be annoying and a student may ask you what it means.
5. If you're typing keys but they're not appearing: you have most likely accidentally pressed `CTRL-s` and froze your terminal. Hit `CTRL-q` to unfreeze.
6. You can always do `:help <command>` to get help on a command.