Student Name: _____ **Lab Section:** _____ **/3** marks

Due Date - Upload to Blackboard by 8:30am Monday March 5, 2012

Submit the completed lab to Blackboard following the Rules for submitting Online Labs and Assignments.

The vi/vim text editor

- **vi** – console terminal-based (non-GUI) text editor, standard version (limited features)
- **vim** – **vi** text editor - improved version (full features)
- **gvim** – **vim** text editor - graphical version (full features) - may not be installed
- **vimtutor** – **vim** text editor interactive tutorial - type at any shell prompt

- The Linux command **vi** is usually a link to the newer **vim** text editor. Some versions of Linux install a smaller, limited version of **vi** as the default editor and you have to explicitly ask for an upgrade to get the full **vim** version with all the great features. **gvim** (if available) starts a GUI-based version of **vim**.
- Some distributions (e.g. Fedora 12) install a smaller, less feature-filled version of **vim** as **vi** (e.g. **/bin/vi**), but then use system aliases to alias **vi** to a larger version of **vim** (e.g. **/usr/bin/vim**), so that you never really know which version you're getting. You can define your own alias to be sure.
- You can find out which version of **vim** you're running using the **:version** and **:help** commands. (Under Fedora, the smaller **vi** version has the wrong help files installed. Use **vim** *not* **vi** on Fedora 12.)

vi/vim : Hard to Learn - Easy to Use

- The **vi** text editor is the **standard editor** available on most **every** Unix-derived system, including Linux, MacOSX, and BSD. It can be used on most **any terminal**, over the slowest of dial-up links. It is a pure console terminal-based program that needs **no mouse** or graphical display screen.
- Similar to learning to touch-type on a keyboard, the **vim** editor is difficult to learn but **easy** to use once you have some mastery of it. Until you master it, you will find using **vi** awkward. Once you master it, you will easily **outperform** anyone using a mouse-based editor such as Notepad. **Learn the tool!**
- The single-character command keys used in **vi** have found their way into **other programs**. The **bash** shell supports a **vi mode** for editing command lines, and the **less** and **more** pager programs (used by the **man** command) use **vi** commands to move around the screen and search for text.
- **vi** is incredibly powerful. Most students refuse to learn it well enough to get out of the **awkward** stage. They never **master** many **vi** commands. They go into **vi** insert mode and perform all text editing using the arrow and backspace keys, turning **vi** into a *slower*, mouseless version of Notepad. If you want a Unix/Linux job, learn the editor tool! If you want to use arrow keys, us the **Pico** or **Nano** editors instead.
- **vi Recommended Reference card #1** (Donald Binder):
  - Front: http://teaching.idallen.com/cst8207/12w/notes/vi_refcard_front.pdf
  - Back: http://teaching.idallen.com/cst8207/12w/notes/vi_refcard_back.pdf
- **vi Reference card #2** (Laurent Gregoire) in multiple formats and languages:
  - http://tnerual.eriogerg.free.fr/vim.html
- **Online interactive web tutorial** (untested): http://lifehacker.com/5844890/the-interactive-vim-tutorial-teaches-you-how-to-use-the-super+efficient-vim-text-editor
- *Vi Lovers Home Page* is http://thomer.com/vi/vi.html

1 Exercise and practice in text editing - Read All The Words

- Do this lab as an **ordinary**, non-root user. Do not run as **root**. Work in your own home directory.
- Open the **recommended `vi`** reference card **before you begin**. All the commands needed for this lab are there, except for "**redo** last command" which is **control-R** in **`vim`** (not available in old versions of **`vi`**).
- **Before you begin**, complete the **`vimtutor`** tutorial program that teaches you **`vim`** basics.
- The following text editing exercise requires **absolute precision**. You are system technicians; small errors in configuration files can disable systems. **Accuracy is important.** You can work on each section of this exercise repeatedly until you get it letter-perfect, then move on to the next section.
- In the underlined spaces below, enter what you typed into **`vi`**/**`vim`** to make the given edit using the **given** number of **`vi`**/**`vim`** command characters (or **fewer**, if you can). Do not use any arrow keys.
- If you make a mistake, simply type the **`vim` undo** command character repeatedly to undo your mistake(s). You can **undo** (and **redo**) multiple times to get the file back to a state you recognize.
- You don't have to save and exit the editor to check your work, below. Use a **second terminal** so that you don't have to leave and re-enter **`vim`** every time. Save your work, then use the other terminal to run the **`file`**, **`wc`**, and **`sum`** commands shown below.
- Do **not** use insert mode and the **arrow keys** to move around the text file. Use **command mode** and the motion commands - they are much faster once you learn them. **Do not use the arrow keys!**
- When below it says "insert a word", it means the word and the **space**(s) around the word, not just the letters of the word. Make sure each word is separated from adjacent words with one space.
- Inserting text always means "insert the text then return to command mode". **Do not remain in insert mode.** Always return to command mode after an edit, so you are ready for the next command.
- Do not enter multiple spaces between words. Do not enter spaces at the start or end of lines.
  - If you want to **see** the extra spaces at the end of lines, enter: **`:set list`**
  - Turn off **list** view using: **`:set nolist`**
- Do not enter extra blank lines, especially blank lines at the bottom of the file. **No extra blank lines!**
- **Save** your work after each **successful** section, so that you can **return** to this point if you don't get the next section correct. Think of these save files as little **snapshots** of your editor session.

1.1 Section save1.txt

1. Create a **`bash`** terminal window that is at least **`80`** columns wide by **`24`** lines long. Larger is good.
2. Start the **`vim`** editor with this (new) file name as an argument: **`lab05.txt`**
3. Turn on **`showmode`** (if necessary) so you know what mode you are in **`:set showmode`**
   (This is enabled by default in Fedora 12.) Also useful (and default on Fedora 12) is: **`:set ruler`**
4. Go into insert mode, enter this single line of text (mouse copy and paste) and **save** your work:
   `Royal Rhonda's repulsive, roaring rabbits ruined Randy's rutabagas`
   - You must only mouse-paste text into **vim** when in **insert** mode. Pasting into **command** mode will run your mouse-pasted text as **vim** commands! Always mouse-paste into **insert** mode!
   - **Case** matters in everything in this exercise. Use **one** space only between each word.
   - There is no final punctuation yet. Those are ASCII apostrophes, not UTF-8 smart quotes.
   - There are no leading or trailing spaces on the one line.
   - There are no leading or trailing blank lines. The file should be exactly one line long.
5. Check your work with **`file`**, **`wc`**, and **`sum`**. You should see these results for your one-line text file:
   ```
   [user@host ~]$ file lab05.txt ; wc lab05.txt ; sum lab05.txt
   lab05.txt: ASCII text
    1  8 67 lab05.txt
   28356     1
   ```
6. When you are successful, make a backup **copy** of your new one-line text file in **`save1.txt`**

1.2 Section save2.txt

1. Continue editing the **`lab05.txt`** file. Let's review some basic commands (see your reference card):
2. With one command keystroke, go to the **beginning** (first character) of the line. That key is: ____
3. Move across the line by **next words**, from left to right. That key is: ____
4. Move back across the line by **beginnings of words**, from right to left. That key is: ____
5. Move across the line by **blank-delimited words** left to right. That key is: ____
6. Move back across the line by **blank-delimited words** right to left. Key used: ____
7. Type one single character to go to the **end of the line** and **simultaneously** enter **insert** mode (one character: ____). Type a space and the word **`today`** after **`rutabagas`**. Do not add any punctuation yet. Remember to leave insert mode. **Never stay in insert mode.**
8. Type one single character to go to the **beginning of the line** and **simultaneously** enter **insert** mode (one character: ____) Type the word **`Oh!`** at the beginning of the line (followed by a space) before **`Royal`**. Remember to leave insert mode. **Never stay in insert mode.**
9. Save and check your work (two more words!) with **`wc`** and **`sum`**: **`1 10 77`** and **`58465`**
10. When you are successful, make a backup copy of your edited one-line text file in **`save2.txt`**

1.3 Section save3.txt

1. Continue editing the **`lab05.txt`** file.
2. Move to the start of the line (one character: ____). **Move forward** to the first upper-case **`R`** (two characters: ____). Efficiently **delete the word `Royal`** (two characters: ____).
3. **Undo** the previous deletion using one character: ____
4. **Redo** the previous deletion using one (control) character: ____
5. Add an exclamation point to the **end** of the line using only three characters. including the character used to get out of insert mode: ____ Remember to leave insert mode. **Never stay in insert mode.**
6. Using only four characters, duplicate the first word in the line. *Hint*: Move to the **beginning** of the line (one character: ____). Yank the **blank-delimited word** into the cut buffer (two characters ____) and **put** the word **before** the cursor position (one character: ____). Make sure you put **before** not after.
7. Save and check your work (ten words) with **`wc`** and **`sum`**: **`1 10 76`** and **`35211`**
8. When you are successful, make a copy of your edited one-line text file in **`save3.txt`**

1.4 Section sve4.txt

1. Continue editing the **`lab05.txt`** file.
2. Turn on line numbers using: **`:set number`**
   The line numbers appear inside **vi/vim** only; they will **not** be saved to the edited file.
3. Using only three or four characters, duplicate the one line in the file **`9`** times; you should end up with **`10`** identical lines in the file. *Hint*: use one or two characters to yank the current line into the cut buffer: ____ then use a repeat count to put the buffer **after** the current line **`9`** times (one digit [the repeat count] and one letter: ____). Make sure you have 10 identical lines. Save the **`10`** identical lines and check your work with **`wc`** and **`sum`**: **`10 100 760`** and **`52495`**
4. When you are successful, make a backup copy of your edited 10-line text file in **`save4.txt`**
5. Continue editing the **`lab05.txt`** file. Using only six characters, yank **all `10`** lines in the file (the whole file) then put it back **`9`** times, giving **100 lines** total. *Hint*: two characters to go to the beginning (first line) of the file: ____ two characters to yank from here to end-of-file into the cut buffer ____ one digit and one character to put the cut buffer **`9`** times **before** the top line: ____ You can type **control-G** to show the current file and number of lines, to confirm that the file is 100 lines. Make sure you put **before** the current line when you put.
6. Using three characters (two digits and a letter), go to line 70: ____
7. Using one character, move to the **top** line on your screen (the top line on your screen is **not** the same as the beginning of the file): ____

8. Type multiple **`k`** to move **straight up**, and watch the screen scroll down one line at a time.
9. Using one character, move to the **middle** line on your screen: ____
10. Using one character, move to the **bottom** line on your screen (the bottom line of your screen is **not** the same as the last line of the file): ____
11. Type multiple **`j`** to move **straight down**, and watch the screen scroll up one line at a time.
12. Use one character to move to the bottom line of the file (move to **end-of-file**): ____
13. Type three characters plus **[Enter]** to **search forward** for the two-letter pattern: **`ru`** ____
14. Type **one** character to repeat the search forward (**in the same direction**) for the next match: ____
15. Move to find match number 50 in the file using two digits (repeat count) and one character: ____
16. Type **one** character to repeat the search backward (**reverse direction**) for the previous match: ____
17. Undo the most recent text change(s) using the undo character: ____ . Your file should be back to **`10`** lines again. (Use **control-G** to check!) Make sure the file has **ten identical** lines before continuing.
18. **Optionally**, turn off line numbering if you don't like it (some people do): **`:set nonumber`**

1.5 Section save5.txt

1. Continue editing the 10-line **`lab05.txt`** file that contains 10 identical lines. (Same as **`save4.txt`**)
2. Go to line **`8`** (use one digit and one letter: ____). Move forward to the first lower-case **`r`** (two characters: ____)
3. Now use six characters to change the word **`repulsive`** to **`ill`** *Hint*: use two command characters to change a word and enter insert mode, type three letters "ill", type one mode-change character: ____ . Remember to leave insert mode. Never stay in insert mode.
4. Move forward to the next lower-case **`r`** on this line (two characters [a repeat move is just **one** character if you know how!]: ____) and type one single character to *repeat the last text-changing command* (one character: ____). Repeating the change will also change the word **`roaring`** to **`ill`** on this line (line **`8`**). You can save a lot of typing using the repeat command.
5. Using only eight command characters, go to line **`6`** and replace the same two words with **`ill`**. *Hint*: go to line **`6`** takes two characters: ____ move forward to **`r`** takes two characters (only **one** if you know how): ____ *repeat the last text change* takes one character: ____ move forward takes two characters (only **one** if you know how): ____ *repeat the last text change* takes one character: ____
6. **Undo** the last change so that **`roaring`** re-appears in line **`6`** (one character undo: ____).
7. **Redo** the last change so that **`ill`** re-appears in line **`6`** (one control-character redo: ____).
8. **Undo** the last change so that **`roaring`** re-appears again in line **`6`** (one character undo: ____).
9. Use nine characters to go line **`4`** and change everything from the start of the line forward to the first comma (**inclusive**) to the word **`How`**. *Hint*: go to line **`4`** takes two characters ____ change text forward to comma takes three characters (a change command followed by a forward motion to a comma - it goes into insert mode) ____ typing **`How`** takes three characters and exit insert mode takes one character.
10. Save and check your work with **`wc`** and **`sum`**: **`10 97 720`** and **`05436`**
11. When you are successful, make a backup copy of your edited 10-line text file in **`save5.txt`**

1.6 Section save6.txt

1. Continue editing the **`lab05.txt`** file.
2. Using five characters (or less), toggle upper/lower case on every character on line two. *Hint*: go to line two takes two characters: ____ toggling **`99`** characters (more than the length of line two) takes two digits and one character: ____ Most of the line will now be upper-case letters. (There is also a **three**-character way to make this same change that works for **any** line length. One of the reference sheets has it.)
3. Use four characters to delete the first four blank-separated words on this line (line two). Hint: one character takes you to the beginning of the line ____ deleting four blank-separated words takes one digit and two characters: ____)

4. Use four characters to replace the only lower-case letter in this line (line two) with upper-case **`C`** so that it says **`CANDY'S`** *Hint*: use two characters to move forward to the **`r`**: ____ use two characters to replace the single **`r`** with **`C`**: ____ Make sure **all six words** on this line (line two) are now upper-case.
5. Save and check your work with **`wc`** and **`sum`**: **`10 93 692`** and **`64996`**
6. When you are successful, make a backup copy of your edited 10-line text file in **`save6.txt`**

1.7 Section save7.txt

1. Continue editing the **`lab05.txt`** file.
2. Use four characters to go to the last line of the file (move to **end of file**) and delete everything from the comma to the end of the line. *Hint*: one character to move to **end-of-file** ____ two characters to move **forward to the character** comma ____ one character to delete from the cursor **to end-of-line** ____
3. Use three characters to make the same change to the line above (to line **`9`**). *Hint*: one character to go **straight up** one line ____ one character to move right one character (onto the comma) ____ one character to repeat the last text change (that deletes to end-of-line) ____
4. Use four characters to make the same change to line **`7`**. *Hint*: two characters to go **straight up** two lines ____ one character to move right one character (onto the comma) ____ one character to repeat the last text change (that deletes to end-of-line) ____
5. Use the same four characters to make the same change to line **`5`**.
6. Use two or three characters to undo the last three text changes, restoring the deleted text at the end of lines **`5`**, **`7`**, and **`9`**: ____ (You can repeat the **undo** command three times, or use a digit **`3`** followed by the undo command letter to repeat **undo** three times.) The text remains missing from line **`10`** only.
7. Save and check your work with **`wc`** and **`sum`**: **`10 87 643`** and **`48878`**
8. When you are successful, make a backup copy of your edited 10-line text file in **`save7.txt`**

1.8 Section save8.txt

1. Continue editing the **`lab05.txt`** file.
2. Use five characters to go to line three and delete the first **`69`** characters, leaving only the word **`today!`** *Hint*: two characters go to line three ____ then use two digits (the repeat count) and one character to delete **`69`** characters moving to the right ____
3. You are currently on line three. Insert the word **`Not`** (and a space) at the start of line three, in front of **`today!`** Remember to leave insert mode. **Never stay in insert mode.**
4. Use three characters to exchange lines three and four. *Hint*: two characters delete the current line three (into the cut buffer; line four moves up and becomes the new current line three) ____ one character puts the cut buffer **after** the current line ____ Line four, containing the contents of the put cut buffer, becomes the current line: **`Not today!`**
5. Starting where you are on line four, use two characters to join line four onto the end of line three. *Hint*: use one character to move **straight up** to line three and one character to **join** the two lines: ____
6. Move line one down to be line three, so that all the lines starting with **`Oh!`** are together. *Hint*: move to line one; delete line one into the cut buffer; move down one line (to line two) and put the cut buffer **after** the current line. Line one is now line three. All lines starting with **`Oh!`** are together.
7. Move to the start of line three, the first line starting with **`Oh!`**
8. Use two characters to delete the first word (including the punctuation) on this line (line three): ____
9. Use three characters to move to the next line (line four) and delete the first **two** words. *Hint*: one character moves **straight down** to the same place on the line below ____ and you can repeat the last text change (delete a word) twice using a digit followed by one character to repeat the last text-changing command ____
10. Use just two characters to delete the first **two** words on the next line (line five) ____ . *Hint*: one character moves **straight down** to the same place on the line below and one character repeats the last text change, which was to delete **two** words.

11. Use just three characters to skip a line and delete the first two words on the next line (line seven). Hint: two characters move **straight down** two lines ____ one character repeats the last change (which was to delete two words) ____ You can save a lot of typing using the *repeat last change* command.
12. Save and check your work with **`wc`** and **`sum`**: **`9 72 551`** and **`54341`**
13. When you are successful, make a backup copy of your edited 9-line text file in **`save8.txt`**

1.9 Hand-in Section - nine file name lines of output

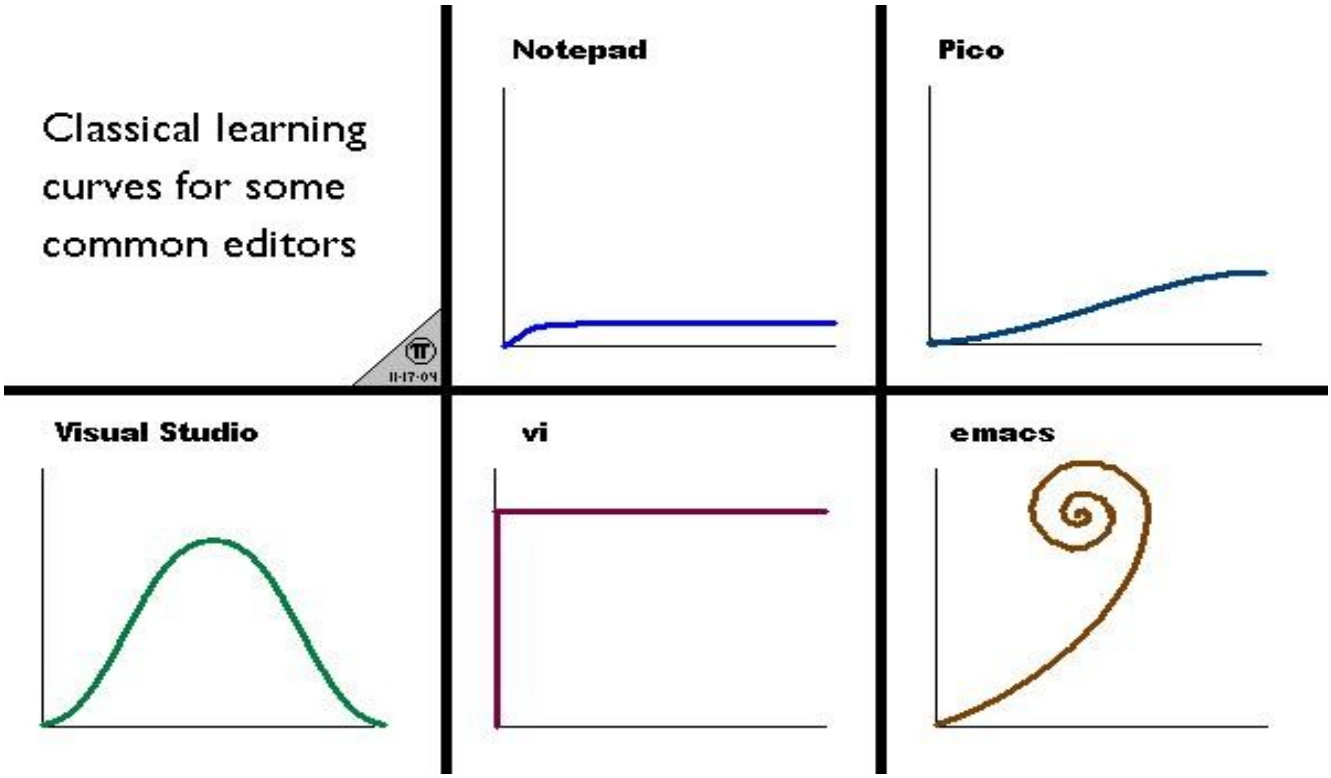
1. Execute the following **two** shell command lines and paste the **nine** lines of output into this page, replacing **all** the **red** the sample output below. You should **cut-and-paste** these two command lines into a **`bash`** terminal window so that you get the typing **exactly** correct. Upload this entire ODT file to Blackboard for marking. Do **not** upload any of the other files. Upload **only** the one ODT document.

```
$ list=" lab05.txt  save?.txt "          # note the two double quotes here and below
$ for f in $list ; do echo "$(sum $f) $(ls -li $f)" ; done
```

Replace **all** this red output with your correct **nine** lines of output from the above two command lines.

12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 lab05.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save1.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save2.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save3.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save4.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save5.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save6.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save7.txt
12345 1 12345 -rw-rw-r--. 1 user user 12345 2011-11-20 00:00 save8.txt

You can delete the graphic image below if you need more space on this page.



2 Advanced Editing Section (optional and cool but *not* for hand in)

- This section shows off two of the more advanced and cool **vi/vim** editor features.
- This section is not for hand-in. I hope it interests you in learning more about **vi/vim**
- Copy the **lab05.txt** file to **adv10.txt** before you begin. Do not edit the wrong file.

2.1 Send any number of lines to an external command

You can send any number of lines from your editor buffer **into the standard input** of any command. The command may process those lines and the **output** of the command will **replace the lines** in the editor buffer.

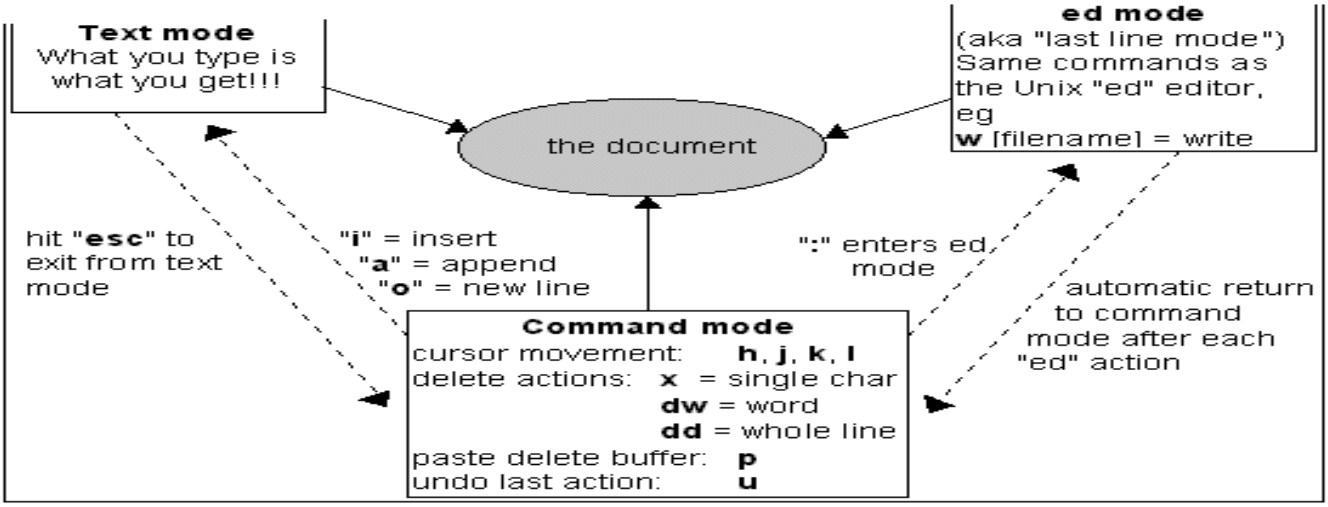
1. Edit the **adv10.txt** file (a copy of **lab05.txt**) that contains nine lines.
2. Go to the first line of the file and type this four-character command followed by **[Enter]**: **!Gwc** ↵
   - this sends all lines from the cursor to end-of-file into **wc** and replaces them with the output of **wc**
   - you should see the expected **9 72 551** output from **wc**
3. Undo the last change. (The output from **wc** disappears and the nine original lines return.)
4. Go to the first line of the file and type this five-character command followed by **[Enter]**: **!Gsum** ↵
   - sends all lines from the cursor to end-of-file into **sum** and replaces them with the output of **sum**
   - you should see the expected **54341** output from **sum**
5. Undo the last change. (The output from **sum** disappears and the nine original lines return.)
6. Go to the last line of the file (line nine) and **open** a new blank line (line ten).
7. On the new blank line type this text: **ls -li lab05.txt save?.txt**
8. Send the current line into **bash** by typing this seven-character command: **!!bash** ↵
   - when you push **[Enter]** this sends just the current line (line ten) into the **bash** shell program
   - **bash** executes the commands it is reading on standard input and the **output** goes into the editor
   - you should see the multi-line output of the **ls** command replacing line ten
9. Go to line one and execute the seven-character command: **!Gsort** ↵ What happens?
10. Go to line five and execute: **!!date** ↵ What happens? Try other commands!
11. Quit the editor without saving anything.

2.2 Record any sequence of edit commands in a macro

You can **record** any complex sequence of editor commands into a single-letter macro that you can **re-execute**.

1. Edit the **adv10.txt** file (a copy of **lab05.txt**) that contains nine lines.
2. Go to the beginning of the top line of the file.
3. Record a macro named **q** by typing **qq** that will turn on the **recording** flag in the bottom line **status bar**. You are now **recording** all the **commands** you are using into buffer **q**. Do these commands:
   - Move forward in this line to the fourth space: **4f** *(there is a **space** after the **f** command)*
   - Append three asterisks and a space after the cursor (and always exit insert mode): **a*** ESC**
   - Close the editing macro by typing a single **q** that turns **off** the recording flag. Macro **q** is ready.
4. The first line now has three asterisks in it: **... CANDY'S *** RUTABAGAS ...**
5. Move to the beginning of any other line in the document and re-execute this macro by typing: **@q**
   Note how the macro executes the same editor commands and makes the same change to this line.
6. Move to the **second** space in any line in the document and re-execute the macro: **@q**
   Note how the macro makes the same change to the **sixth** space, because it moves forward four spaces.
7. Go to the last line of the file (line nine) and **open** a new blank line (line ten).
8. On this blank line dump the macro contents by typing: **"qp** *(that is a **single** double quote to start)*
9. Change the three asterisks to **Fancy Macro** and re-save the macro: **"qyy** *(one double quote)*
10. Go to any line and try the new macro. You can dump and modify any macro after you have recorded it.

Yet Another vi/vim Reference Card (not recommended)



1 Adding new text (switching from command mode into insert mode)

`i`	insert new text in front of cursor	`o`	(lowerc) open new line after (below) cursor
`a`	insert new text after cursor	`O`	(upperc) open new line before (above) cursor
`A`	insert new text at the **end** of the line	`I`	insert new text at the **start** of the line

2 Moving around without the arrow keys (using command mode)

	`k` move straight up (backwards) to the same place one line above		
`h`	move left (backward) one character	`l`	(lower case `L`) move right (forward) one char
`b`	move left (backward) one word	`w`	move right (forward) one word
	`j` move straight down (forward) to the same place one line below		
	↵ (`ENTER`) move down (forward) one line and go to start of line		
`^`	move to the start of the text in a line	`$`	move to the end of a line

3 Editing text (using command mode)

`x`	delete one character (under cursor)	`dd`	cut (delete) entire current line into cut buffer
`r`	replace character (under cursor)	`yy`	copy (yank) current line into cut buffer
`dw`	delete one word (containing cursor)	`p`	(lowercase) put cut/copied text **after**
`cw`	change word (from cursor)	`P`	(uppercase) put cut/copied text **before**
`J`	join the next line to the end of this line	`u ^R`	**undo** and **redo** last editing command

4 Search and Replace

`/text` search forward for next "text"	`?text` search backwards for previous "text"
`:s/old/new/` replace **old** with **new** once	`n N` search forwards, backwards for same text
`:s/old/new/g` replace all occurrences of **old** with **new** on current line only	`:1,$s/old/new/g` replace all occurrences of **old** with **new** on all lines from **1** to **$** (end)

5 Save and/or Exit (also use upper-case `ZZ` in command mode)

`:w filename` write as `filename`	`:q` quit, exit, leave `vi`
`:w` write using existing filename	`:q! quit, discarding changes`