

Lab 3

Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, you can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstance, contact your lab TAs and Jennifer Parham-Mocello.

Pair Programming: In this lab, you can choose a partner for pair programming. **You must be checked off together. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, you will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: [Pair Programming Student Handout](#)

The quiz and linux portion of this lab should be completed independently, but the remainder of the lab can be done using pair programming.

(2 pts) Lab Quiz from Lab #2

1. The following are some of the most commonly used Linux commands. Provide a definition for each.

rm, cp, mv, man

2. Using the above commands, outline the following tasks:
 - Copy contents of one file, assign1.cpp, to another, temp.cpp, in the ~/assign directory, when you are currently in the ~/assign/assign1 directory.
 - Delete a file, temp.cpp, from the ~/assign directory, assume you are in your home directory (~).
3. The following are some of the most common commands in vim. Provide an explanation for each.

:set number

(2 pts) More vim commands

First, create a lab3 directory in your labs directory, and change into the lab3 directory.

```
cd labs
mkdir lab3
cd lab3
```

Now, let's practice using the vim editor following the instructions below. (If needed, refer to Lab #1 for a reference guide to the basic commands)

1. Open a test file for practice – **vim_test.cpp**
2. Save the file using – **:w<Press Return>**
3. Go into insert mode by pressing **i**
4. Type the standard input/output header and main function in your test.cpp file.
5. Get out of insert mode by pressing **<Esc>**.
6. Show the line numbers in vim by typing **:set number**
7. Go to the third line (or whichever line has main) in your code by typing **:3**
8. Delete this line of code by typing **dd**
9. Put the line of code back underneath the first line by moving your cursor to the top line and typing **p**
10. Redo the delete by typing **u** to undo the paste.
11. Paste the line back using **p**
12. Now, copy the line of code by typing **yy** (If you want to copy >1 line, you type one less than the number of lines you want to copy after the y, i.e. y4 copies 5 lines)
13. Paste the line of code by moving your cursor to a line and typing **p**
14. Undo the paste by typing **u**.
15. Try using the **j**, **k**, **h**, and **l** keys to move around the screen. You can also use the arrow keys, but these don't always work.
16. Try searching for "main" in vim with **/main**.
17. Go back into insert mode, **i**, and type some random letters, dkfjdsl, then press **<Esc>** to go back into command mode.
18. Use the **x** to delete all the characters from this random string you typed.
19. Press **i** to begin typing text again.
20. Now, write a small program:
 - Ask the user if she/he likes vi as an editor.
 - Read the 0 or 1 integer value from the user
 - If the user says true, then display a message, "You love vi!"
 - If the user says false, then display a message, "You hate vi!"
21. Now, press **<Esc>** to get back into command mode, and go to the first line in your program by typing **:0**.
22. Then, type **=G** to auto indent your program. You can set the number of spaces you prefer by typing **:set sw=3**, then you want to auto indent again, **=G**. (If you want to auto indent while you type, then use **:set cindent**)
23. You can change your color scheme or background by using one of these commands:
 - :colorscheme evening**
 - :set background=dark**
24. Now, write and quit the file by typing **:wq**

Linux and vim cheat sheet to help you reference some of these commands quickly:
<http://classes.engr.oregonstate.edu/eecs/fall2017/cs161-001/labs/CheatSheet.pdf>

In the next lab you will add some of these useful commands to a vi system resources file.

STOP: You can do Design and Implementation individually or with a partner. Please pair with someone if you feel you are struggling with concepts or simply want to bounce ideas around with someone else!!!

(2 pts) Design

Design is very important when developing programs and there are a variety of ways to approach it. You may draw pictures, write it out in prose or structured text, use pseudo code, and more! The point of design is to give you a blueprint to follow while you are coding. This saves time debugging your program as you can catch mistakes early. It is better to spend one hour of designing than it is to spend five hours debugging.

For this lab you must design a solution to the following problem statement. You will implement your solution!

Problem Statement

A good password has many requirements. Humans have a hard time meeting these requirements left to their own devices. You are tasked with creating a program that will generate a password based on what the user wants in the password.

The user should be able to choose if they want a password with:

- *letters
 - *upper case
 - *lower case
- *numbers

The user should also provide the length of the password and how much of the password should be comprised of their selections. You can generate in order or out of order (see below).

For example:

Welcome to Password Creator!

How long do you want the password to be? 10

Do you want letters (0-no, 1-yes)? 1

Do you want uppercase letters (0-no, 1-yes)? 1

How many characters of the password should be uppercase? 2

Do you want lowercase letters (0-no, 1-yes)? 1

How many characters of the password should be lowercase? 4

Do you want numbers (0-no, 1-yes)? 1

How many characters of the password should be numbers? 4

Your random password is: ACbzdf1254 or AtCde1z254 (either would be acceptable passwords)

Would you like to create another password (0-no, 1-yes)? 0

You need to generate characters and numbers **for** the password length specified by the user. Therefore, you need a way to repeat generating random numbers and characters. You could use a while loop, but it would be more appropriate to use a for loop. For example:

```
for(int i=0; i<pass_length; i++) {  
    //write code you want repeated inside here  
  
}
```

(4 pts) Implementation

Your design must be checked off by a TA before proceeding to code.

In your implementation, you must use the ASCII chart, and you must use rand and srand to generate random values. Think about how we used rand to get a range of numbers in class. Notice characters and numbers are in ranges in the ASCII chart: <http://www.asciitable.com/>

Extended Learning:

Your program should recover from bad input. For example, if a percentage is greater than one hundred or less than zero. The program should recover and prompt for new input until the error has been corrected. The program should then continue as normal.