LAB #2
More Linux, Problem Solving, & C++

Remember, if you and your partner did not finish Lab #1, you need to show your **updated lab to the TAs within 10 minutes of getting to lab**, and you and your partner **will not receive lab credit if you do not get checked off** before leaving each lab. Once you have a zero on a lab, then it cannot be changed because we have no way of know if you were there or not!!!

Reminder: All of our labs involve paired programming. You do not have to keep the same partner for each lab, but you MUST work with someone in each lab!!!

1. **First, find a partner for this lab.** It can be the same partner from Lab #1 or a different partner.

(2 pts) Linux/Vi Practice
These **exercises on the computer need to be repeated by each student** in the pair. This is to ensure that both students understand how to get around in Linux!!!

2. Now, open your secure shell (ssh) client and connect to:
   access.engr.oregonstate.edu

3. This opens a terminal window, which is similar to a Windows command window that presents you with a DOS prompt, but instead of being on your local Windows machine, you are on a remote Linux machine located somewhere else. This is why you are prompted for a username and password to get into the machine because we don't want just anyone on the ENGR machines!!! Once logged into the machine, you get a prompt similar to the DOS prompt, where you can type commands directly to the operating system. In Linux, every command has a manual page, which provides you information on what the command means, how to use the command, and a description of the available options for a command. Linux commands do not have spaces in them and they are lower case. This is important because **Linux is case sensitive**!!! Following a Linux command is a space followed by arguments. Some arguments may be required and some are optional such as options, which are preceded by a dash.

   Linux_command –option required

   If an argument is optional in Linux, then it is enclosed in brackets, [], and required arguments do not have brackets. For example, **man ls**, and notice that everything supplied to ls is optional. You can also use the command and --help to get a brief manual page for the command, i.e. **ls --help**

4. In order to get more familiar with the Linux/UNIX environment, you will do a few Linux-type exercises at the beginning of each lab. Today, we will learn the copy, move, and remove commands, i.e. cp, mv, and rm. First, look at the manual page for each of these commands. **Remember to use the space bar, b, and q for moving around in the manual pages.
   man cp
   man mv
   man rm

5. First, let's play with these new commands before moving forward. Copy your hello.cpp program from the labs/lab1 directory to your home directory, **cp labs/lab1/hello.cpp ~**. This says, copy the hello.cpp file located in the labs/lab1 directory into my home directory. Use **ls** to list the directory contents and make sure you have a hello.cpp file in your home directory.

6. Now, rename the file to hello2.cpp by using the move command, **mv hello.cpp hello2.cpp**. Use **ls** to list the directory contents and make sure you no longer have a hello.cpp file and you now have a hello2.cpp file.

7. Create a test directory, and then change into that directory.
   mkdir test
   cd test

8. Copy the hello2.cpp file from your home directory to the test directory you are currently in. **Remember that **..** is up/back a directory. You could also say mv ~/hello2.cpp . because you know that hello2.cpp is in your home directory.
   cp ../hello2.cpp .

9. Now, go back to your home directory or up/back a directory, and remove the file hello2.cpp file in your home directory and remove the test directory and its contents, which contains the file hello2.cpp. Use ls to make sure you see the hello2.cpp file and the test directory in your home directory.
   cd ..
   ls
   rm hello2.cpp (when prompted press n so you don't remove it)
   rm –f hello2.cpp (notice no prompt, -f forcefully removes without asking)
   rm test (notice it won't remove a directory, even with -f)
   rm –r test (notice the prompt, -r recursively descends into a directory to remove it and its contents, note you can use –rf together to avoid all the prompts)
   ls (you shouldn't see hello2.cpp or test)

10. Change into your labs directory, create a lab2 directory, and then change into that directory. **DO NOT use spaces in directory or file names in Linux.
    cd labs
    mkdir lab2
    cd lab2

11. There are a few shortcuts in Linux that you want to be familiar with using. One is the use of up/down arrows to move through your **history of commands**. At the shell prompt, press the up arrow and note what happens, and then press the down arrow and note what happens.

12. Another useful shortcut is **tab completion**. Go up two directories with **cd ../..**, and then let's change back into the labs directory. This time, after typing cd and l, then press the tab key. This will complete your labs word because it is the only option in your home directory that starts with an l. Now, try changing into the lab2 directory again using tab completion, but this time you'll be presented with two options that start with an l.

(4 pts) Read User Input/Wrtie C++ Program

13. **Problem Statement:** A metric ton is 35,273.92 ounces. In order to get the weight of a package of breakfast cereal in ounces from the user, you need to use **std::cin**. Since this only reads information from the user, we need to print a message telling the user what to enter, before we read the data. For instance:

```
#include <iostream>

int main() {
   float cereal_weight;

   std::cout << “Enter the weight (in ounces) of a box of cereal: “;
   std:: cin >> cereal_weight;   //cereal_weight contains input

   return 0;
}
```

Now, you can calculate the weight in metric tons, as well as the number of boxes needed to yield one metric ton of cereal. As a pair, write the C++ program based on your design from Lab #1. In addition, get the expected lifespan and boxes of cereal eaten by the user in a week, to find out how many boxes of cereal the user eats in a lifetime.

Input:	weight of a package of breakfast cereal in ounces
	expected lifespan of the user
	number of boxes of cereal eaten in a week
Output:	the weight of the cereal in metric tons
	number of boxes needed to yield one metric ton
	number of boxes eaten by user in a lifetime.

(2 pts) Self-Reflection/Think about your thinking

14. At this point, you and your partner need to address these metacognitive questions by either typing or handwriting your answers:

    Planning:
    Why do you think we are learning to design first, then program?
    What resources did you use to complete this lab?
    Did you use your time wisely to finish the lab on time?
    Monitoring:
    What was the most difficult and confusing part of this lab?
    What did you do to address the difficulties and confusions?
    Evaluation:
    How well do you think you accomplished the lab?
    If you had to do this over, what would you have done differently?
    What worked well for you and what didn't?

(2 pts) Design Steps for Solution

15. As a pair, **design/write out the steps on a piece of paper** that you need to go through to solve the following problem.

    Problem Statement: A government research lab has concluded that an artificial sweetener commonly used in diet soda will cause death in laboratory mice. A friend of yours is desperate to lose weight but cannot give up soda. Your friend wants to know how much diet soda it is possible to drink without dying as a result. Write a program to supply the answer. The input to the program is the amount of artificial sweetener needed to kill a mouse, the weight of the mouse, and the weight of the dieter. To ensure the safety of your friend, be sure the program requests the weight at which the dieter will stop dieting, rather than the dieter's current weight. Assume that diet soda contains one-tenth of 1% artificial sweetener. Use a variable declaration with the modifier const or #define to give a name to this fraction. You may want to express the percentage as the double value 0.001.

    You and your partner write down how long you think you will live and how many diet sodas you drink per week. Determine if you will ever drink the amount of diet sodas that would cause death☺