Lab 1

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 Instructor.

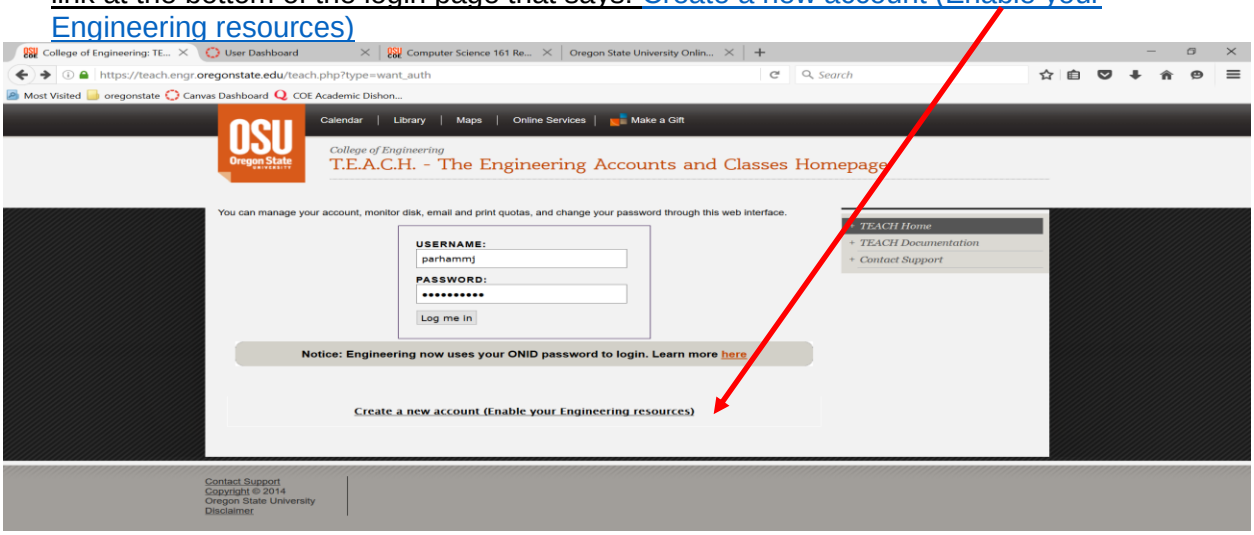
(2 pts) Icebreaker: Believe it or not, we do have to communicate with others being a computer scientist and electrical engineer, and others have to communicate with us. Let's get to know our peers. Get into small groups of 4-5, and answer the following about everyone in the group:

- Do you have any experience programming in C/C++? If so, how long?
- What did you like (or dislike) the most about your winter break?
- Establish one goal for yourself this Winter quarter.

(2 pts) Getting Set Up

These **exercises need to be completed individually by each student**. This is to ensure that everyone gets setup properly! You can certainly ask your neighbor or friend questions! ☺

1. **If you do not have an ENGR account**, then you will need to create one by clicking on the link at the bottom of the login page that says: Create a new account (Enable your Engineering resources)



2. After you have an engineering account, set up your terminal using the below instructions.
   Windows: http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/labs/windows_instructions.pdf
   Mac/Linux: http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/labs/mac_linux_instructions.pdf

3. After launching your terminal, at the prompt, type the following commands to look at your files and directories in your home directory. Note the differences to your lab instructor.
   ls

ls -a
ls -l
ls –al

**Note: You should notice a . and .. directory listed. The . directory refers to your current directory, and the .. directory refers to one directory above your current directory. The ~ refers to your home directory.

4. UNIX/Linux provides you manual pages for the commands. You are encouraged to read these manual pages when you have questions about a specific command or want more details about the options to use with a command. Use the space bar to scroll forward through the manual pages (one page at a time), press b to scroll backwards (one page at a time), and press q to quit the manual page. You can also use the up and down arrow keys to scroll forward and backward one line at a time, but who wants to do this☺
   man ls

5. In addition, if you are not sure what a command is in UNIX/Linux, then you can find the appropriate command using apropos and a keyword. For example, what are the UNIX/Linux commands for editing a file, working with a directory, etc.
   apropos editor
   apropos directory

6. You may have noticed that you get more text than what can fit in your terminal window. To view data a page at a time in your terminal, you can pipe the command contents through another command called less. This will allow you to scroll through the pages using space bar, b, and q just as you did with the manual pages.
   apropos directory | less

(2 pts) Create Directories, Write C++ Program, & Answer Metacognitive Questions

7. Make a directory in your home directory named labs, and change into the labs directory.
   mkdir labs
   cd labs

8. Create a directory in your labs directory named lab1, and change into the lab1 directory.
   mkdir lab1
   cd lab1
9. Make a note of your present working directory.
   pwd

10. You can go back/up a directory by using two periods/dots together, and you can go back to your home directory by using the tilde, ~. Use the pwd to confirm you are back in your home directory.
    cd ..
    cd ~
    pwd

11. Now, change into the labs/lab1directory by using your up arrow key to take you through the history of commands you've used in the past. You should see the cd labs and cd lab1 command you typed earlier. You can also change directly into the lab1 directory by using cd labs/lab1.

12. A good rule of thumb is to use pwd at any time to determine where you are, in case you forget. Also, don't be scared to use ls as often as you need to see a listing of your current directory.

13. Use the vim editor to create a C++ file containing your first C++ program.
    vim hello.cpp

    Here is an overview of vi/vim and a set of useful commands:
    http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/labs/unix_cheat_sheet.pdf

14. Write the infamous "hello world" program as your first piece of C++ code. Use the style guideline on the class website for suggestions on how to format your code:
http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/assignments/161_style_guideline.pdf

```
#include <iostream>

int main() {

   std::cout << "Hello World!" << std::endl;

   return 0;
}
```

15. Compile and execute your C++ "hello world" program.
    g++ hello.cpp –o hello
    ./hello

For each question, make a change to the hello.cpp file, and recompile (g++ hello.cpp –o hello) and possibly execute the program (./hello) if there are no errors to answer the question

Question 1: What happens when you forget the semicolon at the end of the cout statement? Be specific.

```
std::cout << "Hello World!" << std::endl
```

Question 2: What happens when you forget to include iostream, i.e. remove `#include <iostream>` from the program? Be specific.

Question 3: What happens when you remove the << std::endl? What is the difference in having it and not having it?

```
std::cout << "Hello World!";
```

(2 pts) Add a variable and reading input to the program:

16. Now, let's add a variable (or a place to store information) in our program. We will create a place to store an integer (or whole number). Then, we'll ask/prompt the user to enter a number, read the input from the user, and display it back to the screen.

```
#include <iostream>

int main() {
   int number;

   std::cout << "Hello World!" << std::endl;

   std::cout << "Enter an integer whole number: ";
   std::cin >> number;
   std::cout << number << std::endl;

   return 0;
}
```

17. Compile and execute your C++ "hello world" program again with the new changes.
    g++ hello.cpp –o hello
    ./hello

18. Logout of the remote machine by typing
    exit (or logout)

For each question, make a change to the hello.cpp file, and recompile (g++ hello.cpp –o hello) and possibly execute the program (./hello) if there are no errors to answer the question

Question 4: What is the difference between the first and second program you wrote?

Question 5: What happens if you mix the direction of >> with the cin? Be specific.

```
std::cin << number;
```

Question 6: Why do you need to create a variable to display a number entered by the user?

(2 pts) Upload Written Document and Program (.cpp) to TEACH

Since you have to upload your assignments from your local computer or the engr server to the TEACH website, then we should practice now. It is easy to upload files from your local computer to TEACH, as long as you remember where you save it. However, it is not as easy for you to upload files from the ENGR server to TEACH.

Transfer Files

You can transfer the file to your own computer, then upload it on TEACH. Return to the appropriate instructions for details:

Windows: http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/labs/windows_instructions.pdf

Mac/Linux: http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/labs/mac_linux_instructions.pdf