# Lab 01: Getting Started

**Assigned**: *Wednesday, June 24th, 2020*
**Due**: *Tuesday, June 30th, 2020*
**Points**: *100 (normalized to 100 in gradebook)*

- There is NO MAKEUP for missed assignments.
- We are strict about enforcing the LATE POLICY for all assignments (see syllabus).

## Introduction

Your first lab for this week is an introduction to programming on CSIL and the tools you'll need for this course. The intended outcomes are:

- Learn about the tools you will be using in this class viz Gradescope.

This lab must be completed INDIVIDUALLY. In the subsequent labs you are encouraged to work with your programming partner.

## Get setup with the tools for this course

### Create a CoE account if you don't have one already

We encourage you to complete all programming assignments by logging in to the machines in the Computer Science labs, or to connect remotely. To do this you will need a **College of Engineering account**. You can create an account online at https://accounts.engr.ucsb.edu/create.

If you are enrolled in *any* CoE course this quarter, you can create your account immediately. If you are not, you will need to contact the ECI Help Desk at help@engineering.ucsb.edu.

### Get setup with Gradescope

We will use Gradescope to grade all your homework, exams and lab/programming assignments. You should have received information about how to log into Gradescope.

Log into our class site on https://www.gradescope.com/: **CS16 Fall 2020** and navigate to the **lab01** assignment. Keep this page open to submit your code at the end of the lab.

## Implement and submit a simple C++ program

### Step 1: Open a Terminal and write 2 programs: "Hello World" and "Calculate"

The first step in every assignment will be to open a **terminal window**, which will be the environment you use to write, compile, and run your programs.

If you are working on your laptop, whether Windows, Mac or Linux, the instructions below will tell you how to connect to a CSIL machine.

Connect to one of the following machines:

- `csil-01.cs.ucsb.edu`
- `csil-02.cs.ucsb.edu`, etc.
- etc...
- through `csil-48.cs.ucsb.edu`

Do not connect to `csil.cs.ucsb.edu`, instead connect to `csil-N.cs.ucsb.edu` (where ***N*** is a number between 01 and 48). You'll get much better performance on those individual machines, because they are much less heavily loaded and have newer hardware, as compared to `csil.cs.ucsb.edu`.

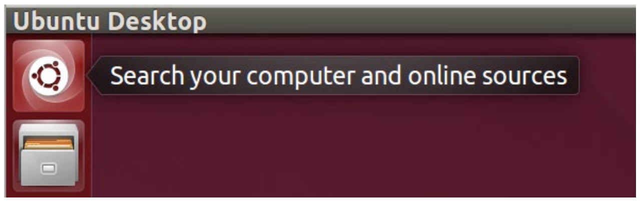
- If you are working on your laptop and it is a Mac or Linux machine, go to **Step 2a**.
- If you are working on your laptop and it is a Windows machine, go to **Step 2b**.

## Step 2a: Connecting to CSIL via SSH on Mac OS X or Linux

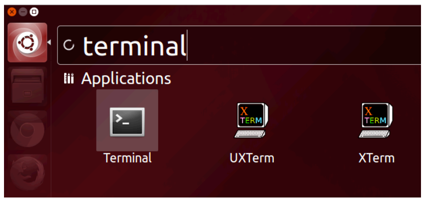
To get started on Mac OS X or Linux, you first need to open a terminal program. This involves slightly different steps on either OS.

*On Ubuntu (an example of a Linux OS):*

1. Find the search menu. It appears at the top of the Unity bar:

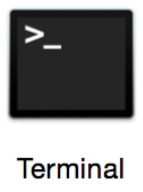


1. Click on that icon to open the search menu. Then type "terminal" and click on the "Terminal" application which appears:



*On Mac OS X:*

1. Open the "Terminal" application. It is found inside the *Applications* folder of your main drive, inside the *Utilities* subfolder. The icon looks like this:



You can also find it using **Spotlight** (i.e. **Command-Space**) by typing "terminal" and pressing ENTER.

Once you have a terminal window open on your machine, you next need to **connect to the CSIL server remotely**.

You will do this using a UNIX command (an internet protocol, really) called *SSH* (short for Secure Shell).

Type the following command in your terminal, replacing **USERNAME** with **your CoE username**:

```
$ ssh USERNAME@csil-15.cs.ucsb.edu
```

Note that we are using **csil-15** as an example here (you can connect to any of the 48 CSIL machines). SSH will first ask you a question which looks like this:

```
The authenticity of host 'csil-15.cs.ucsb.edu (128.111.43.14)' can't be established.
RSA key fingerprint is 90:ab:6a:31:0b:81:62:25:9b:11:50:05:18:d3:1a:b5.
Are you sure you want to continue connecting (yes/no)?
```

Type **yes** and then ENTER to continue.

It will next ask for your CoE account password. When you type it in, nothing will show on the screen (not even dots). However, what you type is still being sent and once you are finished with your password, you can press ENTER to login.

**You should now be remotely connected to CSIL!** You can make sure by typing the following command (which will tell you what machine you are currently issuing commands to):

```
$ hostname
```

This should show **csil-15.cs.ucsb.edu**. You can now do anything you could normally do in a terminal window in CSIL or the Phelps lab (except run graphical programs).

## Extra Note: Graphical Forwarding

This is not required or necessary to use CSIL remotely, so if you are not interested, go ahead and skip this part.

If you have an X windows system installed you can get graphical applications running by *forwarding* X from CSIL to your machine. To do this, add the **-X** option to the SSH command like this:

```
$ ssh -X USERNAME@csil.cs.ucsb.edu
```

X windows is almost always installed on graphical Linux, and can be installed on Mac OS X as XQuartz (which can be found at http://xquartz.macosforge.org/landing/).

## Step 2c: Connecting to CSIL via SSH on Windows

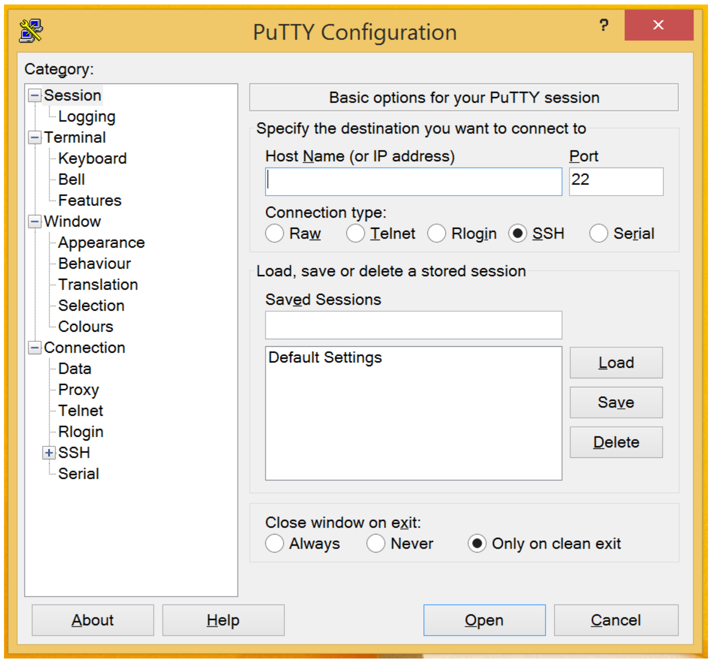
### 1. with PuTTY (if you have non-Windows 10)

To connect remotely on Windows machines, especially if you don't have Windows 10, we recommend using a program called *PuTTY*. This program is a well-known and widely-used SSH client for the Windows OS. This works on new and old versions of Windows.

First, download the program from **http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html**. You only need the executable file **putty.exe**, but feel free to download any other programs that you want. The page includes portable versions and a version with an installer. *Always make sure to download PuTTY from this site*, so that you can make sure it is the correct program.

Once downloaded, run PuTTY like you would open other programs. If you just download the **putty.exe** file, you can open it from your downloads folder directly. You can also move it to any other location on your machine and open it from there. If you used the installer, open PuTTY from the Start Menu.

When PuTTY opens, you should see a window that looks like this:



Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

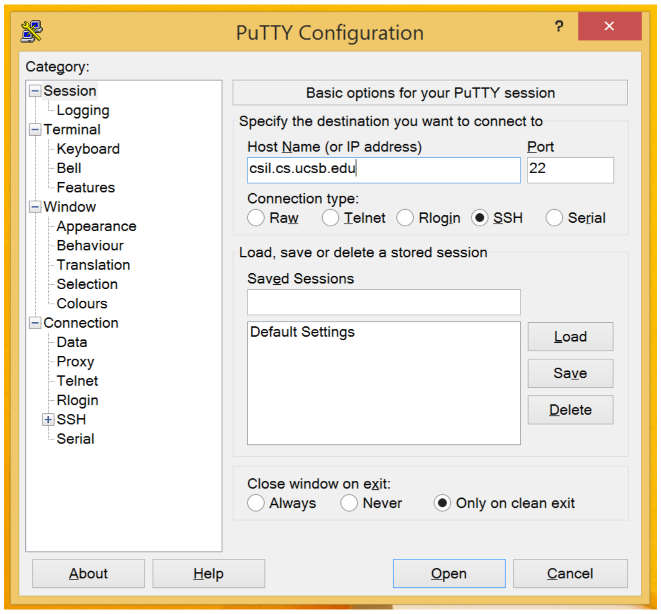
Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

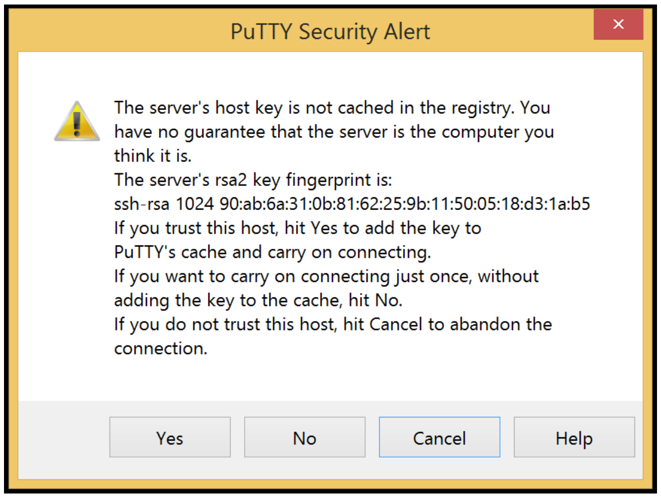
Type **csil-15.cs.ucsb.edu** into the box labeled "Host Name (or IP address)". Note that we are using **csil-15** as an example here (you can connect to any of the 48 CSIL machines).

Leave the "Port" setting at **22** and leave the "**SSH**" button checked. The window should now look like this:



Specify the destination you want to connect to
Host Name (or IP address): csil.cs.ucsb.edu
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

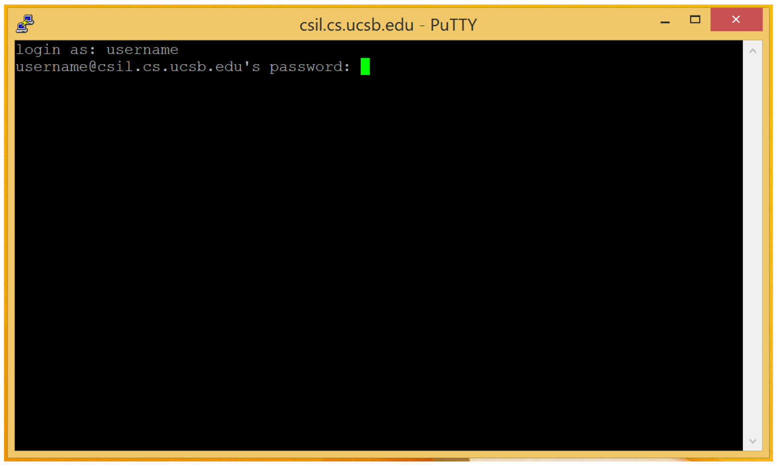
Then click on the "Open" button to connect. PuTTY will then show a prompt which looks like this:



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 1024 90:ab:6a:31:0b:81:62:25:9b:11:50:05:18:d3:1a:b5
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Click "Yes" to accept and have PuTTY remember CSIL's key.

Once a connection is made, CSIL will ask for both your username and then your password. Type in your CSIL username and password. The password will not be shown on the screen, but the characters you type are being used. This step will look something like this (with your username instead of "username"):



```
login as: username
username@csil.cs.ucsb.edu's password:
```

Once you have logged in successfully, you should be connected remotely to the CSIL server. Run the following command to make sure (this command shows the full host name of the machine you are logged in to):

```
$ hostname
```

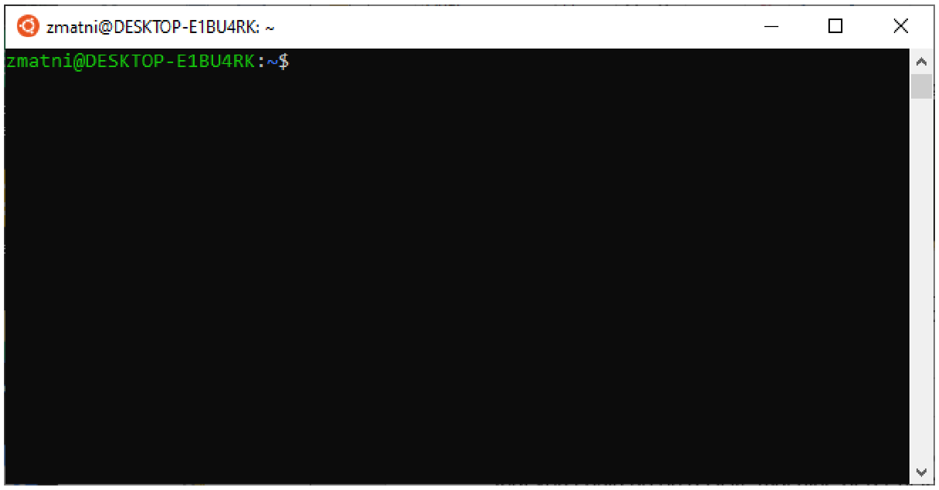
This command should output **csil-15.cs.ucsb.edu**. You can now do anything in this terminal window that you could do on a CSIL machine or a Phelps lab machine, except run graphical applications.

If you want to run graphical applications, instead of using puTTY, you can use other programs, such as **MobaXterm**. This is not necessary for this class (so completely optional for you).

## 2. with Linux Bash Shell (Windows 10 only)

Instead of using PuTTY, the Windows 10 operating system has a built-in Linux terminal (like the MacOs does), but you will need to install it once before you use it. This is better than PuTTY, but it only works on Windows 10.

The website at this URL: https://itsfoss.com/install-bash-on-windows/ has an easy step-by-step process to tell you how to install and enable Linux on your Windows 10. When you run it after that, it's just like running any application/program on your computer and it will look like this:



```
zmatni@DESKTOP-E1BU4RK:~$
```

Type the following command in your terminal, replacing **USERNAME** with **your CoE username**:

```
$ ssh USERNAME@csil-15.cs.ucsb.edu
```

Note that we are using **csil-15** as an example here (you can connect to any of the 48 CSIL machines). SSH will first ask you a question which looks like this:

```
The authenticity of host 'csil-15.cs.ucsb.edu (128.111.43.14)' can't be established.
RSA key fingerprint is 90:ab:6a:31:0b:81:62:25:9b:11:50:05:18:d3:1a:b5.
Are you sure you want to continue connecting (yes/no)?
```

Type **yes** and then ENTER to continue.

It will next ask for your CoE account password. When you type it in, nothing will show on the screen (not even dots). However, what you type is still being sent and once you are finished with your password, you can press ENTER to login.

**You should now be remotely connected to CSIL!** You can make sure by typing the following command (which will tell you what machine you are currently issuing commands to):

```
$ hostname
```

This should show **csil-15.cs.ucsb.edu**. You can now do anything you could normally do in a terminal window in CSIL or the Phelps lab (except run graphical programs).

## Step 3: Create cs16 and lab01 directories

Now that your environment is set up, you next will need to create a directory (a folder is also called *directory* in Linux) that will contain all your work for the course. Then, inside that directory, you will need to create another directory to contain your work for this assignment.

To create your CS16 directory, use the **mkdir** command. Type the following in the terminal and press enter:

```
$ mkdir cs16
```

The **$** represents the terminal prompt; *you won't type this character*. Whenever you see it, that means that the following command is intended to be typed into the terminal window and run by pressing enter.

You can see list of files and directories in the current directory with **ls** command. Type the following in the terminal and press enter:

```
$ ls
```

You should be able to see the directory you just created i.e. **cs16**

Now move into that new CS16 directory with the **cd** command as follows:

```
$ cd cs16
```

And create and move into a lab01 directory:

```
$ mkdir lab01
$ cd lab01
```

At any time, you can check what directory you are current in with the command **pwd**. It will output the full path of the current directory. For example, if you are inside your **lab01** directory, you might see:

```
/cs/student/yourcsilname/cs16/lab01
```

Knowing how to navigate a UNIX environment and issue UNIX commands is VERY valuable to computer scientists and engineers. To learn more UNIX commands, there are lot of cool Web resources and books on the topic. This is one website I found that's a good introductory page: **Useful unix commands** (http://mally.stanford.edu/~sr/computing/basic-unix.html)

## Step 4: Editing text files for programming

Let's take a little detour on how to best create and modify text files. These will carry all the code (regardless of computer language) that we want to assemble, compile, and execute.

You are surely all familiar with Microsoft Word as a widely-used "word processor", but please DO **NOT** USE MS WORD TO WRITE PROGRAMS!!! :)

Instead, for programming, you have access to a very large number of excellent text editors - most of them are free to use! I will introduce you to just 4 of them below. If you already have a favorite editor and know how to use it well, then you don't have to change and use something else, just for this class.

In fact, *AND PLEASE NOTE THIS*, no one editor is necessarily "better" than another. It is a matter of your preference. This is a great time for you to explore multiple options and then pick one. Once you pick an editor of choice, STICK WITH IT!

As you progress in your Computer Science education and, subsequently, your careers in CS, make sure you end up learning how to use more than one editor. You can still have a "favorite" that you excel at using, but at least have a working familiarity with others.

1. **vim** for UNIX-based OS

vim (or sometimes called vi) is a popular text editor that's also available on just about every UNIX machine (including the ones that you're using in the CS labs) and UNIX-based machines (like MacOS computers).

To run vim on a UNIX machine or a MacOS machine, open up a terminal (see above for how to do that on Macs) and type:

```
$ vim
```

To edit a file (let's say it's called "filename"), you'd type:

```
$ vim <filename>
```

To customize your vim environment for a better coding experience with C/C++ copy this .vimrc file from the instructor folder to your home folder using the following command:

```
cp /cs/faculty/zmatni/public_html/cs16f20/.vimrc ~/
```

Again, to learn how to use vim, there is no substitute for PRACTICE!!! Again, there are multiple online resources that you can look at and here are some of them:

- **About vim** (http://www.vim.org/about.php)
- **vim commands - a handy reference card** (http://tnerual.eriogerg.free.fr/vimqrc.html)
- **another reference cheat sheet for vim** (https://www.fprintf.net/vimCheatSheet.html)

2. **Sublime Text 2** for Windows OS and MacOS X — see **the product website** (it's a program that you'd have to download)

3. **Notepad++** for Windows OS — see **the product website** (it's a program that you'd have to download)

4. **emacs** for UNIX-based OS

emacs is a very popular editor that's available on just about every UNIX machine (including the ones that you're using in the CS labs) and UNIX-based machines (like MacOS computers).

To run emacs on a UNIX machine or a MacOS machine, open up a terminal (see above for how to do that on Macs) and type:

```
$ emacs
```

To edit a file (let's say it's called "filename"), you'd type:

```
$ emacs <filename>
```

Go ahead and edit a file named "hello.cpp"

In the editor type a few comments on the first line as follows:

```
        // hello.cpp
        // Your name
        // Your perm number
```

Note that in C++, you can insert comments by preceding the comment with `//`

To save the file, press Ctrl-x, Ctrl-s. To exit emacs and return to the UNIX shell, press Ctrl-x, Ctrl-c

To learn how to use emacs, there is no substitute for PRACTICE!!! Of course, there are multiple online resources that you can look at (especially given emacs' popularity) and here are some of them:

- **emacs commands - a handy reference card** (https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf)
- **a beginner's guide to emacs** (http://www.jesshamrick.com/2012/09/10/absolute-beginners-guide-to-emacs)

## Step 5: Create and edit your 2 files containing C++ programs

Now it's time to write the programs! If you're comfortable with one of the reviewed text editors, then go ahead and use one. Otherwise, here are some **emacs hints** (https://ucsb-cs16-s18-mirza.github.io/lab/lab00/emacs_hints/) and some **vim hints** (https://ucsb-cs16-s18-mirza.github.io/lab/lab00/vim_hints/)

### hello.cpp

This part of the assignment only needs you to write a program that prints out two lines on the display, and nothing else. Remember that newlines, spaces and tabs are characters. **The output should look exactly as follows** (no space before or after each line, except the 2 newlines):

```
Hello, world!
I am ready for CS16!
```

Start with a "skeleton program" (or template) that contains the necessary structure but that does not do anything:

```
#include <iostream>

using namespace std;

int main() {
    // Your printing code should go here

    return 0;
}
```

Go ahead and type this in to the **hello.cpp** file. Alternatively, you can copy and paste it directly from this page. Next, you will need to replace the comment with code to print out the expected output. Comments in C++ are lines that start with **//** or text between **/*** and ***/**. The second type can span multiple lines.

Important note: For students familiar with Python, remember that lines starting with the **#** character are not comments in C++. Rather, they are important include lines that allow your program to use the input and output functionality. Make sure to copy those lines in your program as well.
Only **//** or **/*** create comments in C++.

To print out text to the terminal, you can use the **cout** stream. To output something use the **<<** operator as shown below:

```
cout << "This will be printed out to the terminal" << endl;
```

The **endl** command will cause a newline (i.e. a carriage return) to be printed and the next print to go on the next line.

You can adapt this line to achieve the objective of the assignment. **Remember that we need to print two lines, each with a newline at the end.** You can do this with one or two statements.

MAKE SURE THAT YOU SAVE THE FILE WITH THIS CODE AS **hello.cpp** (case-sensitive).

**calculate.cpp**

This second part of the assignment needs you to take 3 integer values as standard input from the user, assign them to 3 variables (call them **var1**, **var2**, **var3**), and finally calculate *and* print the calculation of: **3*(var1 + 2*var2) – 4*var3**. The 3 variables are integers (so they can be positive or negative whole numbers).

**The output should look exactly as follows when using the following example.** Of course, different user inputs should yield different answers. Note that the user inputs are bolded for your convenience (they won't actually show up as bolded on the screen). There are newline characters after each line.

```
Please enter 3 numbers.
2
3
4
The formula result is: 8
```

Start with the same template (skeleton program) as the previous exercise.

MAKE SURE THAT YOU SAVE THE FILE WITH THIS CODE AS **calculate.cpp** (case-sensitive).

## Step 6: Compile the Code

Now that the code is written, we need to *compile* it. This will be done using a special program called a *compiler*.

Before moving on, **make sure you save your code** and close the text editor. The following step will be done in the terminal.

For C++ code we will use the **g++** compiler that's built-into many UNIX machines (it even works on most MacOS terminal programs). You can compile the **hello.cpp** file into an executable called **hello** with the following command:

```
$ g++ -o hello hello.cpp
```

This will compile your code (hello.cpp) and make an executable version of it (hello). Specifically, it will tell the compiler to take the source code file **hello.cpp** and compile and link it to an executable called **hello**.

If the compilation is successful, you won't see any output from the compiler, but if you issue a UNIX **ls** command, you should see a new file has appeared: one called **hello**.

You can then use the following command to **run** your program:

```
$ ./hello
```

Which means "in the current directory, as represented by the **.** character, run the program **hello**". You should then see the program output the two expected lines.

The other possibility is that the program may **not compile successfully**. What to do then?
If you run the **g++** command and are unsuccessful with your compilation, then you might see an output that looks like this:

```
hello.cpp: In function 'int main()':
hello.cpp:10:1: error: expected ';' before '}' token
         }
```

The compiler will try to give you hints on the line (in this case, it's complaining about line 10) where the error occurs, and also about what the error is (in this case a missing semicolon). You will also note that, in this case, an output executable file is not produced.

If you encounter an error, use the compiler hints and examine the line in question. If the compiler message is not sufficient to identify the error (which happens more than sometimes), you can search online to see when the error occurs in general. Once you have fixed the error, run the compilation command again. De-bugging a program code is a necessary ritual in almost all programs written (even those written by expert coders). More on that in a later class.

Now, do the same for **calculate.cpp**.

## Step 7: Submit your programs for grading

Once you are satisfied that your programs are correct, then it's time to submit then.

Log into your account on https://www.gradescope.com/ and navigate to our course site: **CS16 Fall 2020**. Select this assignment. Then click on the "Submit" button on the bottom right corner to make a submission.

You will be given the option of uploading files from your local machine or submitting the code that is in a github repo. Choose the *first* option and follow the steps to upload your 2 source files: **hello.cpp** and **calculate.cpp** to Gradescope (do not upload any other files). If your files do not have those exact names, the autograder will not work (and you won't get any points). You can resubmit your files as many times as you like before the assignment deadline.

You should receive 50 points for each correct program (for a total of 100 points).

Congratulations on completing your first C++ programs! ☺
You can log out using the **exit** command in UNIX:

```
$ exit
```