Lab 1: Released Wednesday, week 1 and due Tuesday, week 2

Getting Started

Important overall note: If the instructions say to run `some text in monospace font`, this means to type that text into your command line (often called Terminal) and hit enter to run it. If you need to run a command in a particular directory, I will mention where that is, otherwise any directory is fine. Refer to https://sites.cs.ucsb.edu/~maradowning/cs16/refs/quickcmdline.html for some quick start instructions on how to use the command line.

I often use <fillertexthere> to indicate that this is a field you should fill out with your own information, like <username>. Do not include the <> when running the command.

Step 1: C++ Setup

There are a handful of ways you can choose to set up your coding environment, feel free to choose one or more here.

1. CSIL (preferred)
2. Install C++ on your personal computer

CSIL setup:
Create a College of Engineering (CoE) account if you do not have one already: https://accounts.engr.ucsb.edu/create.

If you are on Mac or Linux:
ssh into CSIL: run (in Terminal)
`ssh <username>@csil.cs.ucsb.edu`
Enter your password when prompted. This command will bring you to your home directory on CSIL.

If you are on Windows:
Run the same command as you would in Mac/Linux, in Command Prompt.

When you want to exit CSIL, just run `exit` and the connection will be closed.

This is the preferred option because it allows you to have a pre-set-up C++ environment (CSIL already has g++ installed) and does not vary based on personal operating system. If you are having problems, this will also be the easiest option for the TAs to help with, since having a controlled coding environment rules out one major way things could be going wrong. Working on a remote server (using the command line) is also a super valuable skill in CS.

This is also the only setup option I expect the TAs to be ready to help with. If they can help you with the others that's great, but most likely this will only be possible if they have personal experience with your operating system.

If you want to work on CSIL without using vim, take a look at these guides:

- https://ucsb-cs16.github.io/topics/csil_mount_csil_drive_via_macos/
- https://ucsb-cs16.github.io/topics/csil_mount_csil_drive_via_windows/

Installing and using C++ locally:
If you are on Mac or Linux, first run
`g++ --version`
In the command line. If the message you get gives you a version number, and that version number is 4.8 or later, you are done!

- In general, if you're not sure if you have a command line program installed, running `<programname> --version` is a great way to figure it out without running it

If on Linux (and you don't already have g++ 4.8 or later installed):
Run:
`sudo apt-get update && sudo apt-get upgrade -y`
Then
`sudo apt install g++`

If on Mac (and you don't already have g++ 4.8 or later installed):
You will first need to install XCode, using the app store. It is a pretty big package, so this could take some time.

Secondly, you will need to install homebrew. Run:
`/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
Next run:
`brew install gcc`

If on Windows:
First follow the WSL2 setup guide at https://docs.microsoft.com/en-us/windows/wsl/setup/environment, then use the Linux instructions above.

One useful tip for working with WSL2: If you want to have your files easily accessible via your normal Windows setup, you can open a folder where you want to work and then Shift-rightclick to open a list of options, choose "Open linux shell here". This will open a unix command line in that folder, so when you create files using the command line they will be visible (and editable) from your Windows folder.

Step 2: Github setup

Note: if you have not yet completed HW1, please do so now. If you have completed HW1 but don't have access to the course Github yet, please reach out to the course staff to get added. This process may not be immediate.

Installation:
If you have chosen to work on CSIL, the command line tool git will already be installed. Do not attempt to install anything on CSIL yourself (it won't let you).

If you have decided to work on your own computer, you will need to install git on your computer. First run
`git --version`
to check if you already have it.

Linux (and WSL) instructions:
run
`sudo apt-get install git`

Mac instructions:
git should already be installed if you installed XCode above, let me know if this is not the case.

SSH Key Generation:
Follow the guide in: https://ucsb-cs156.github.io/topics/github_ssh_keys/
There are a few changes I need to note in this guide:

- Why would I want to do that? section: the new answer is because you cannot use HTTPS on a private repository
- Start from "What's a public/private key pair?", end at (before) "What if I already cloned with HTTPS"
- The CSIL instructions will work on your personal computer provided you are using Mac, Linux, or WSL as instructed above
- If this does not work for whatever reason, come talk to the course staff!
    - Github changes their SSH key requirements quite often, there is a (small) chance this guide will be outdated by the time you start using it

If you are working on CSIL, this step should be done on CSIL. If you are not, this step will happen on your own computer.
If you plan to use multiple computers, you will need to complete this step for **every** computer you plan to work on.

Git Configuration:
Git will want to know who you are when you try to push/pull from Github. Run
`git config --global user.name “yourUsernameHere”`
And
`git config --global user.email “your.email@here”`
To make sure it knows who you are. I would really appreciate it if you used your Github username and email so I’m sure that any updates to your code actually come from you. When you push updates to Github, the username you put here will be shown as the author of the update.

Cloning the Github Repository (Finally!):
For the first lab, we will all use the same “clone” command, since we are not going to push our own changes to the code. We really just need to get it onto our computer.
The “clone” command will create a folder with all of the files for the lab (currently stored on Github). This folder will be created within the folder you run the command in, so choose wisely. You cannot move an entire github folder to a new place on your computer without severing the connection between the files and Github—while this doesn’t really matter for this lab, in the future it will prevent you from “pushing” your changes to Github, which is how I will see them and be able to grade your labs.
Once you have chosen a good place, run
`git clone git@github.com:ucsb-cs16-1-u22/Lab01.git`
You will see that after this command, there is a new folder in this directory named Lab01. If you look inside that folder you will see the files for this lab.
This is the only Github instruction you will use this week.

Step 3: Run your first C++ programs!

Program 1: compiletest.cpp
This program doesn't require you to do any coding—it's just to get you used to compiling and running a C++ program. You'll be asked to record the output of this program for your Gauchospace submission.

Look in the file named compiletest.cpp within your Lab01 folder. The contents should be as follows:

```
#include <iostream>
using namespace std;

int main(){
    cout << "Welcome to CS16!" << endl;
    return 0;
}
```

Inside the Lab01 folder, run
`g++ -std=c++11 -Wall compiletest.cpp -o compiletest`
to compile this program. If you check the contents of the folder again with `ls`, there should be a new file called compiletest (no extension).

From here, run
`./compiletest`
to run your compiled code. Only one line should print—record this line in your Gradescope submission.

Grading: 15 points

Program 2: hello.cpp
This part of the assignment 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 spaces before or after each line, except the 2 newlines):

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

If you look in the file hello.cpp, you will find a "skeleton program" (template) that has the necessary structure, but none of the printing code, for this program.

```
#include <iostream>
using namespace std;

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

    return 0;
}
```

Edit this file to print the expected lines. After this, run
`g++ -std=c++11 -Wall hello.cpp -o hello`
to compile your program. Make sure there are no errors or warnings. Then, run
`./hello`
to check that the output prints as expected.

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 keyword will create a newline (i.e. a carriage return).

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

You will submit the line(s) you added to the program to Gradescope.

Grading: 35 points

Program 3: calculate.cpp
This part of the lab needs you to:

- take 3 integer values as standard input from the user (using cin)
- assign them to 3 variables (call them var1, var2, var3)
- and calculate and print the result of the formula: 3 * (var1 + 2 * var2) - 4 * var3.

The output should look exactly as follows when using the following example input values.

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

There is an empty file named calculate.cpp for this program, and you can start by copying in the same template as the previous exercise.

To compile this program, run
`g++ -std=c++11 -Wall calculate.cpp -o calculate`
Make sure there are no errors or warnings.
Run
`./calculate`
to run your program. I suggest testing it out with at least 4 sets of numbers (including some zeroes and negative numbers) to make sure it works properly.

You will submit the entire contents of this program to Gradescope.

Grading: 50 points

- Code compiles: 5 points
- No compiler warnings: 5 points
- Code passes all tests: 40 points

Step 4: Gradescope Submission

You're (almost) done with the first lab! The final step is just to submit your work on Gradescope. Unlike the rest of the labs this quarter, this one will have a few more questions than usual so you can submit your coding work without dealing with Github. By next week you'll submit code only through Github, and the lab submission on Gradescope will be just to indicate that you're done.