

# COMP 3522 Lab #1

Christopher Thompson, Jeffrey Yim  
jyim3@bcit.ca

Due Friday 11:59pm

## Welcome!

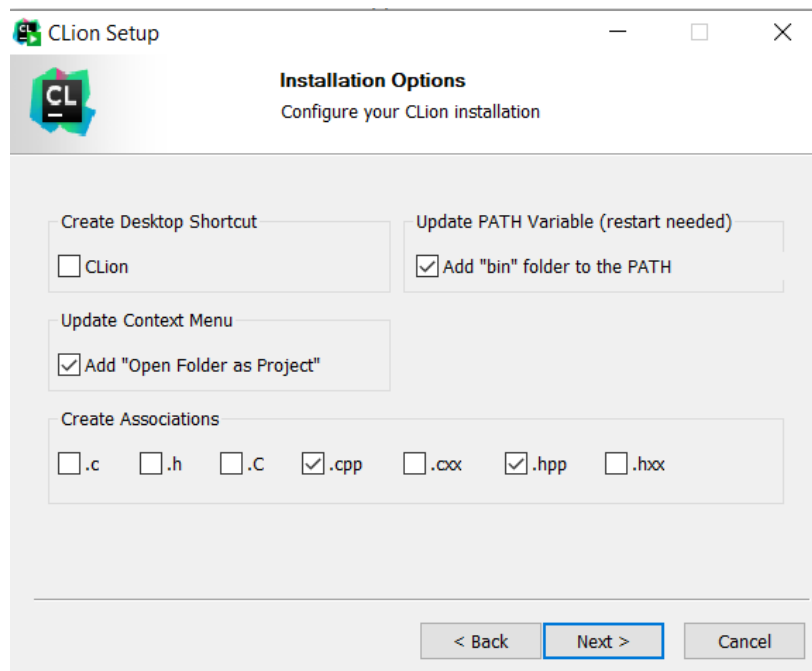
Welcome to your first COMP 3522 lab. Today's lab is all about setting up your tools, and using them to generate and share a simple Hello World application. You will **install CLion, git, g++, and create your first C++ project**.

You'll also learn how to use Github classroom, commit and push your code. Let's get started!

## 1 CLion setup

Please complete the following:

1. Start by signing up for a free renewable one-year JetBrains student license at <https://www.jetbrains.com/student/> so you can download and use any of the JetBrains desktop products.
2. Install the JetBrains C++ IDE called CLion (2023.1.5) on your machine <https://www.jetbrains.com/clion/download/>.
  - Install with the following options:



3. **Ensure git is installed on your laptop. If it isn't, install it now from** <https://git-scm.com/downloads>. Continue installing with all default settings using the installer
4. Sign up for a free GitHub Student Developer pack at <https://education.github.com/pack> so you can store unlimited private repositories on GitHub.
5. We will compile using **g++** this term. If you are using Windows, you will need to install Cygwin (<https://www.cygwin.com/>) or MinGW (<https://www.mingw-w64.org/>).
6. If you're having issues installing Cygwin on PC, try following this Youtube video: <https://youtu.be/IVNLaGkvDw0>. Make sure to install the following packages:

- **gcc-g++**

View Pending Search gcc-g++ Clear

Package	Current	New	Src?	Categories	Size	Description
gcc-g++	11.3.0-1	11.4.0-1	<input type="checkbox"/>	Devel	18,186k	GNU Compiler Collection (C++)

- **make**

View Pending Search  Clear

Package	Current	New	Src?	Categories	Size	Description
make	4.3-1	4.4.1-2	<input type="checkbox"/>	Devel	584k	The GNU version of the 'make' utility

- **gdb**

View Pending Search gdb Clear

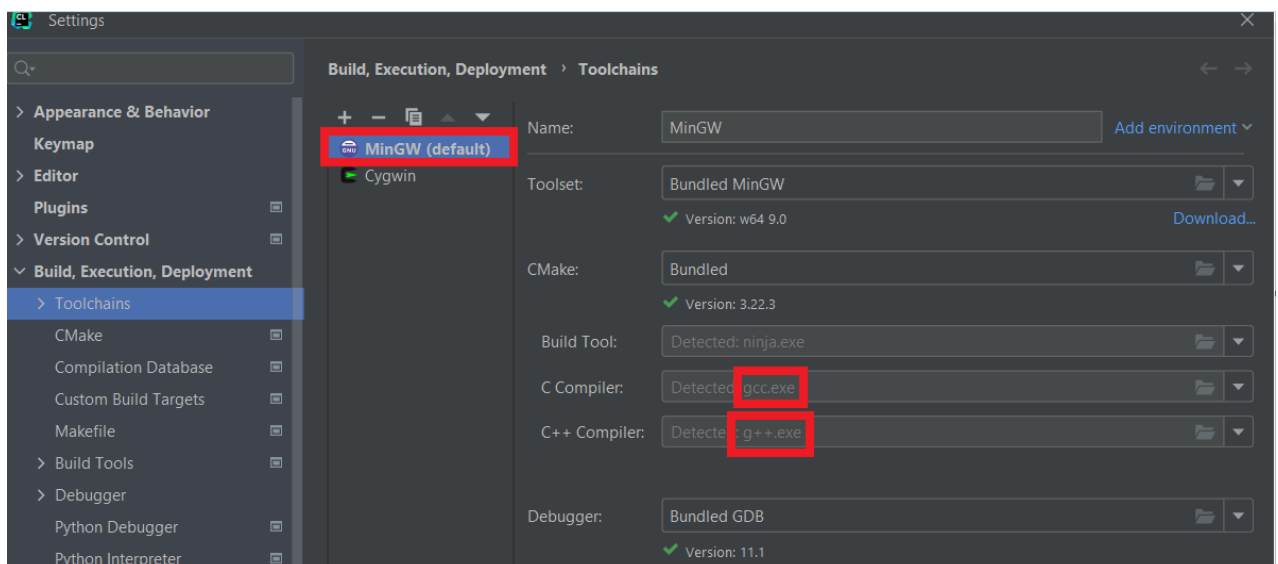
Package	Current	New	Src?	Categories	Size	Description
gdb	10.2-1	12.1-1	<input type="checkbox"/>	Devel	3,606k	The GNU Debugger

- If you are using a Mac, the g++ command invokes the LLVM compiler (if you don't believe me, open a Terminal and enter the command `g++ -version` (that's a double dash)). The easiest way to fix this is to install g++ using Homebrew, and then invoke g++ using the command `g++-13`. Be aware the `g++` version may be newer on your machine:

#### Command Line

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
$ brew install gcc
$ g++-10 --version
```

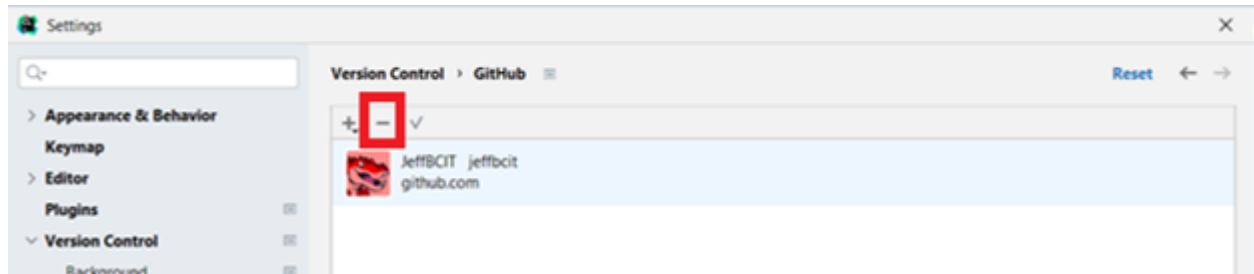
- Ensure CLion uses the correct **g++ compiler** in Settings/Preferences by selecting Build, Execution, Deployment | Toolchains and setting C++ Compiler.
- If Cygwin isn't working for you, try using MinGW. Just ensure the C++ Compiler is using **g++**



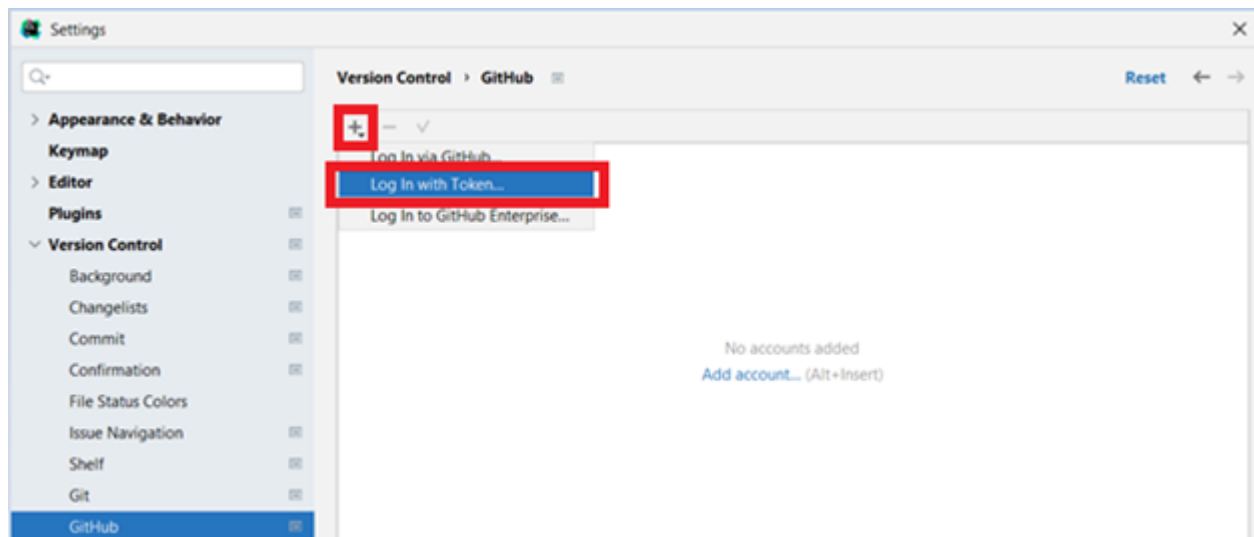
# Getting the Personal Access Token

You may need to get a Personal Access Token before cloning the repo

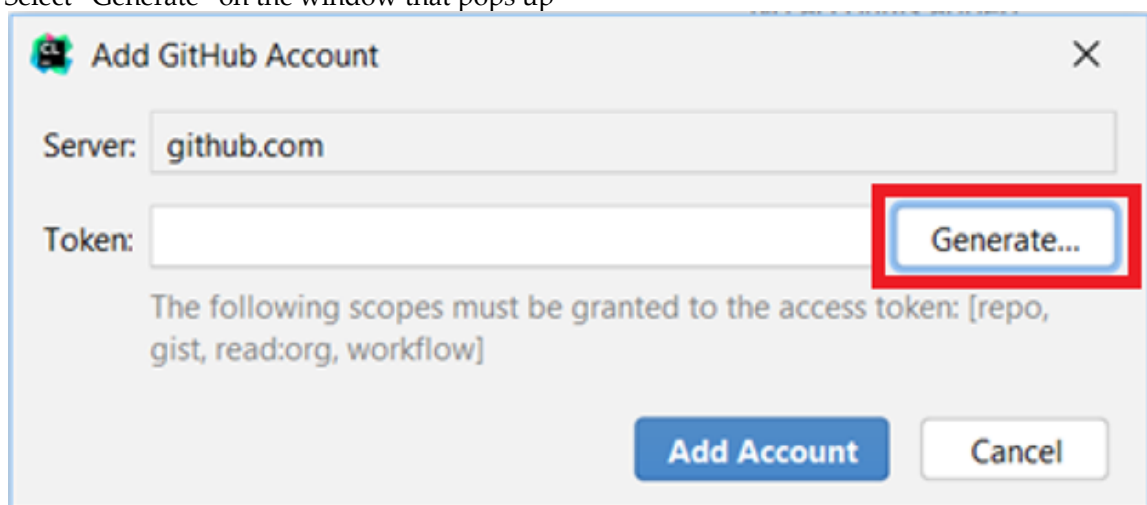
1. To log in with a github personal access token, open **CLion** and go to Settings > Version Control > Github. If you're already logged in, press the – button to log out



2. Select the + icon, then select “Log in with Token...”



3. Select “Generate” on the window that pops up



4. A web page should open up mentioning, “New personal access token”. **Change the token expiration to something past the end of the term**

## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

CLion GitHub integration plugin

What's this token for?

### Expiration \*

90 days



The token will expire on Wed, Sep 28 2022

5. Scroll down to the bottom and press “Generate Token”

<input checked="" type="checkbox"/>	<b>gist</b>	Create gists
<input type="checkbox"/>	<b>notifications</b>	Access notifications
<input type="checkbox"/>	<b>user</b>	Update ALL user data
<input type="checkbox"/>	read:user	Read ALL user profile data
<input type="checkbox"/>	user:email	Access user email addresses (read-only)
<input type="checkbox"/>	user:follow	Follow and unfollow users
<input type="checkbox"/>	<b>delete_repo</b>	Delete repositories
<input type="checkbox"/>	<b>write:discussion</b>	Read and write team discussions
<input type="checkbox"/>	read:discussion	Read team discussions
<input type="checkbox"/>	<b>admin:enterprise</b>	Full control of enterprises
<input type="checkbox"/>	manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/>	read:enterprise	Read enterprise profile data
<input type="checkbox"/>	<b>admin:gpg_key</b>	Full control of public user GPG keys ( <a href="#">Developer Preview</a> )
<input type="checkbox"/>	write:gpg_key	Write public user GPG keys
<input type="checkbox"/>	read:gpg_key	Read public user GPG keys

Generate token


Cancel

6. Copy the generated token value from the website and paste it into CLion and select "Add Account". You should now be able to clone from your repo

igs

## Edit personal access token

Make sure to copy your personal access token now. You won't be able to see it again!


**Your token value here** 

**Note**

CLion GitHub integration plugin

What's this token for?

No accounts added

 **Add GitHub Account** ×

Server:

Token:

The following scopes must be granted to the access token: [repo, gist, read:org, workflow]

## Getting the Course's sample code:

1. We'll be using **Github Classroom** this term to track your lab/assignment submissions. For now, let's use it to clone all of the course's sample code. Go to this link: <https://classroom.github.com/a/guzoC6DD> . And **Accept the assignment**:

## Accept the assignment — Course sample code


Once you accept this assignment, you will be granted access to the `course-sample-code-jeffbcit` repository in the **BCIT-COMPUTING** organization on GitHub.

Accept this assignment

2. Refresh the following page and click on the repository link that's provided

You accepted the assignment, **Course sample code**.

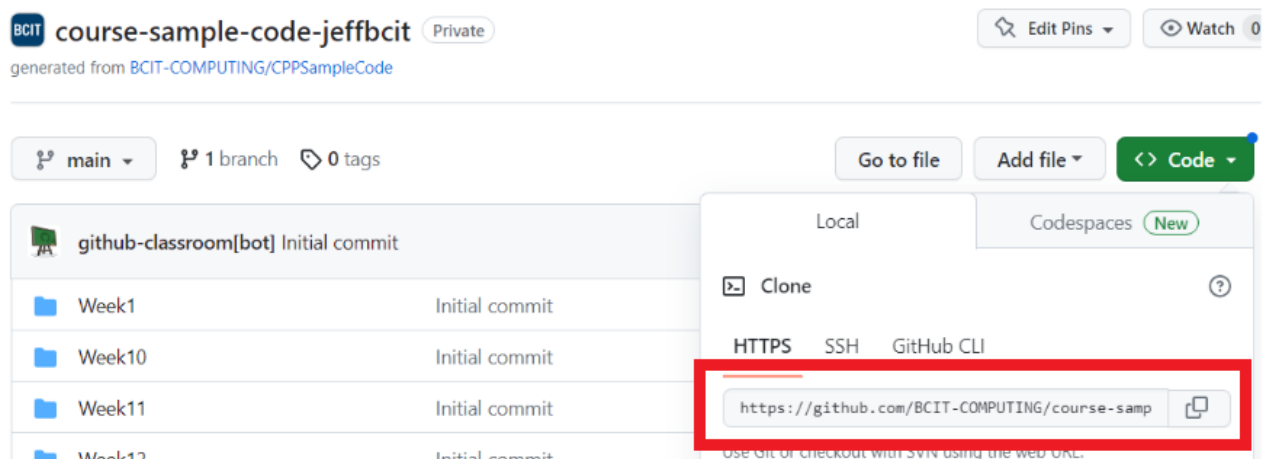
Your assignment repository has been created:

 <https://github.com/BCIT-COMPUTING/course-sample-code-jeffbcit>

We've configured the repository associated with this assignment ([update](#)).

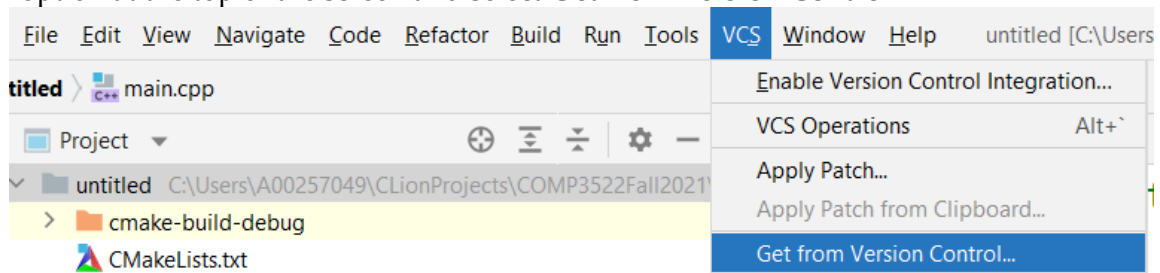
## Clone from repo to Clion

3. Next you will clone the repository. Click the **green Code button** and copy the repo url

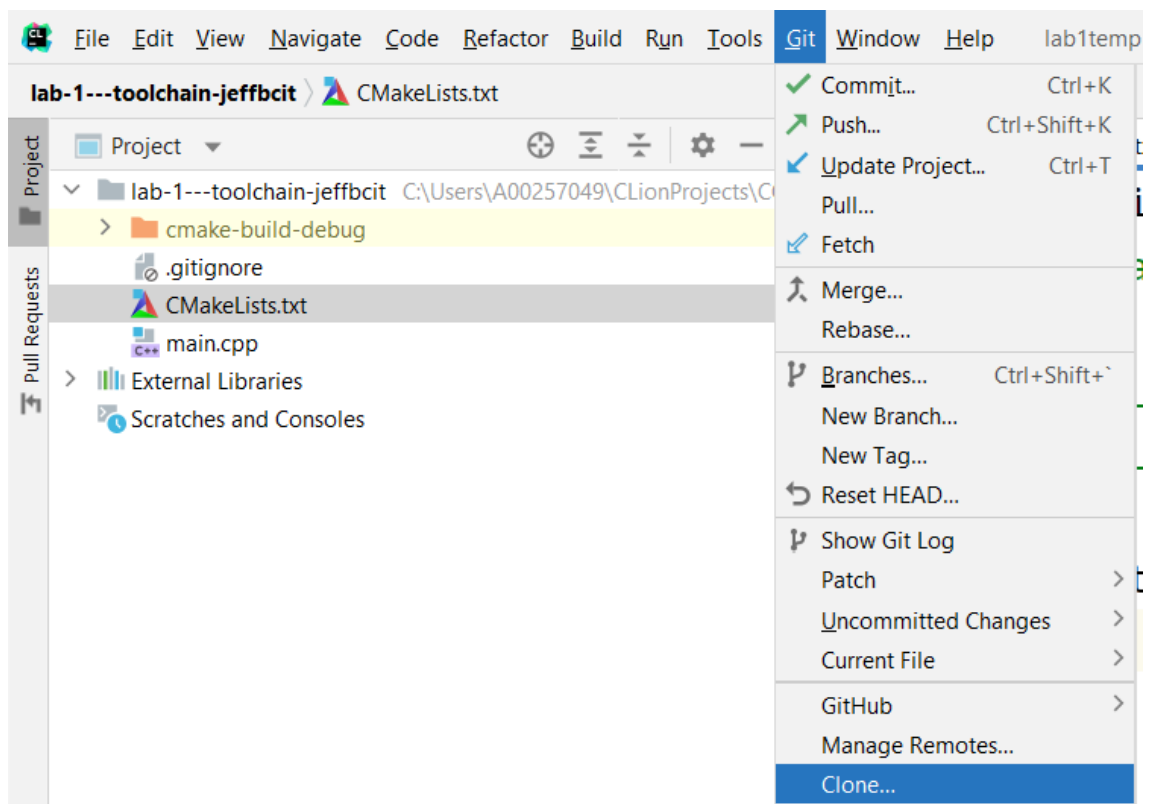


The screenshot shows the GitHub repository page for `course-sample-code-jeffbcit` in the `BCIT-COMPUTING` organization. The repository is marked as 'Private' and 'generated from BCIT-COMPUTING/CPPSampleCode'. The 'main' branch is selected, with 1 branch and 0 tags. The 'Code' button is highlighted in green. The 'Clone' dropdown menu is open, showing options for 'Local' and 'Codespaces' (marked 'New'). Under 'Local', the 'Clone' option is selected, and the 'HTTPS' method is chosen. The repository URL `https://github.com/BCIT-COMPUTING/course-samp` is displayed and highlighted with a red box, with a copy icon to its right. Below the URL, it says 'Use Git or checkout with SVN using the web URL.'

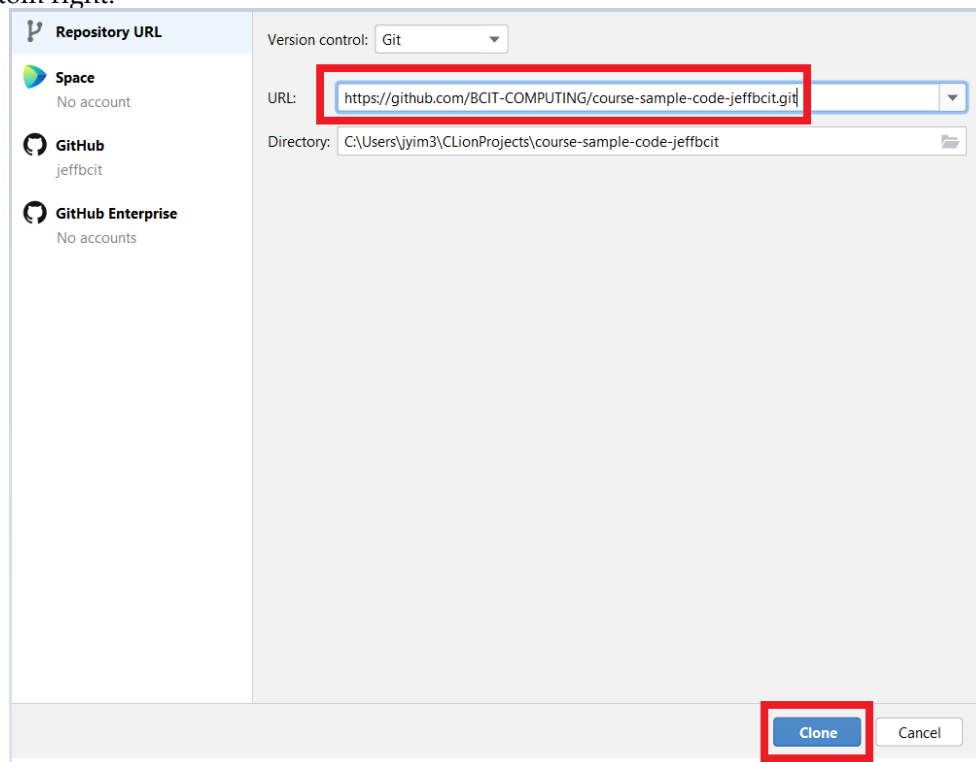
4. Return to CLion and create a new project, or resume an existing project. Find **VCS** option at the top of the screen and select **Get from Version Control**



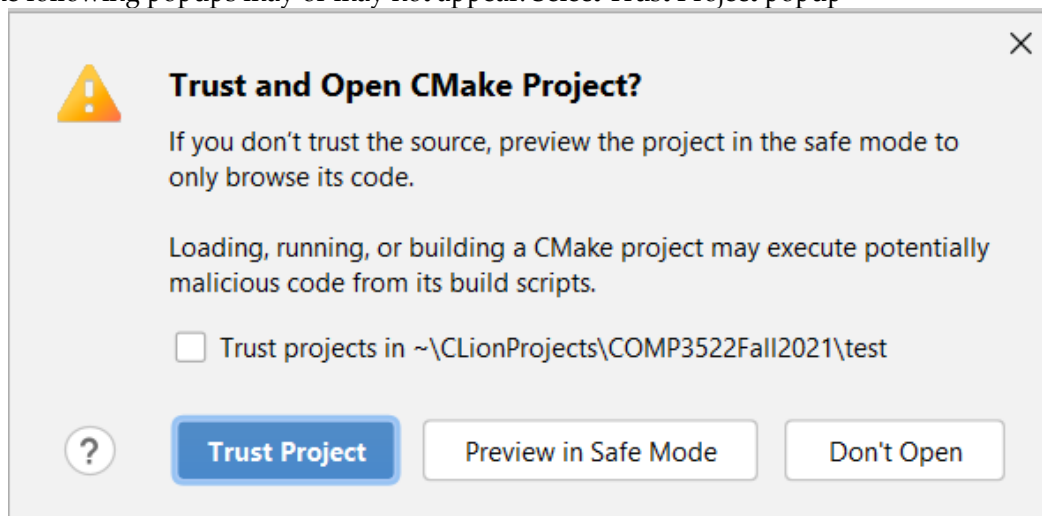
Alternatively, if you don't see the **VCS** option, you may see **Git**. In that case select the **Clone...** option



5. Paste the repo link into the **URL** field of the window that appears and select **Clone** at the bottom right.

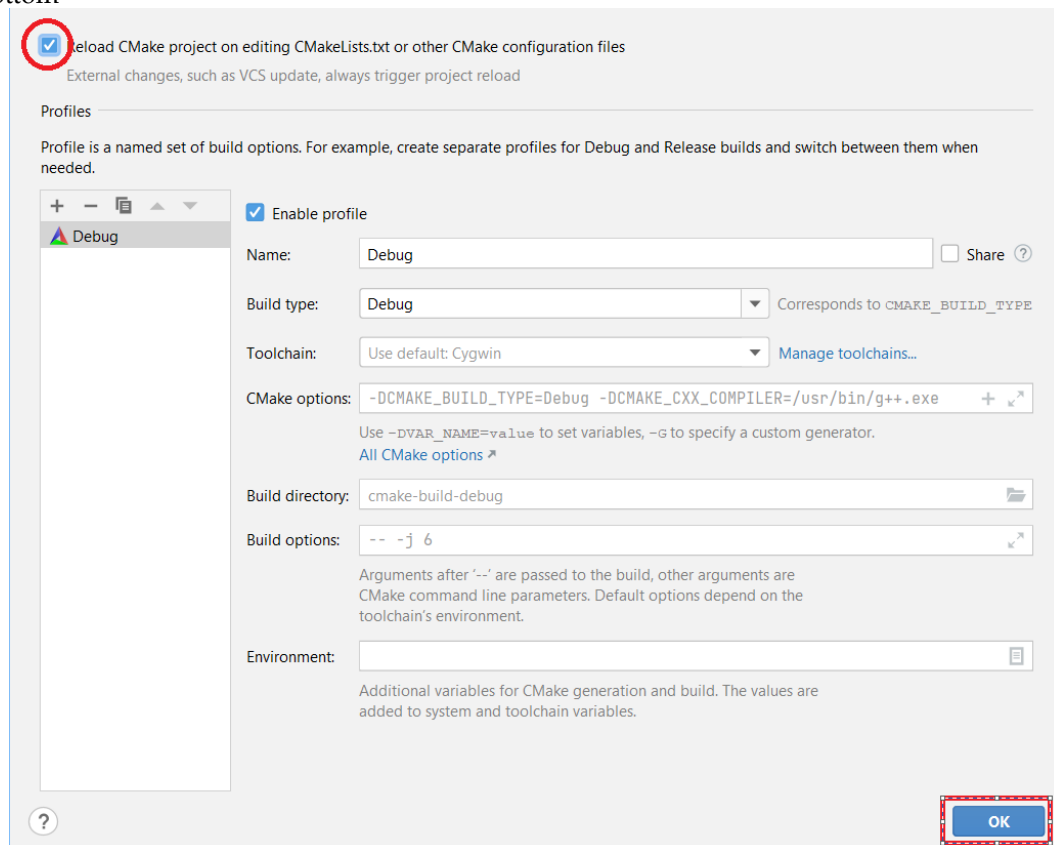


6. The following popups may or may not appear. Select Trust Project popup



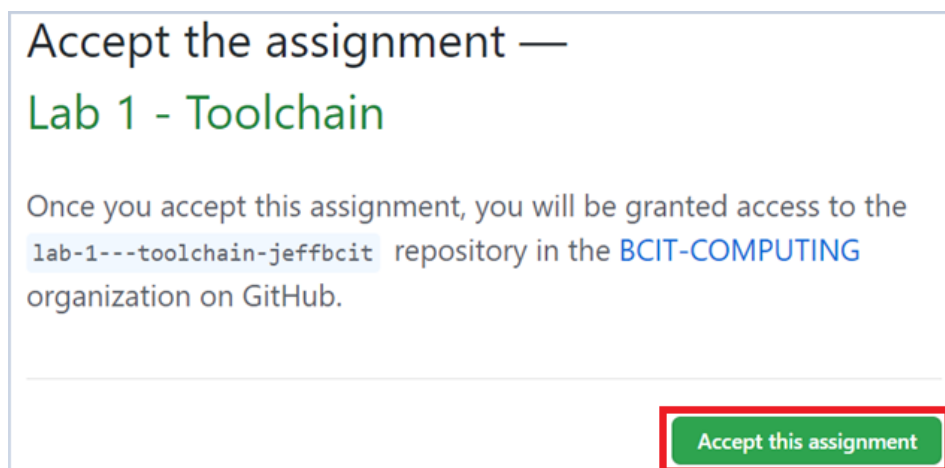


7. Check the box in the top left asking about reloading CMake projects. And select **OK** at the bottom

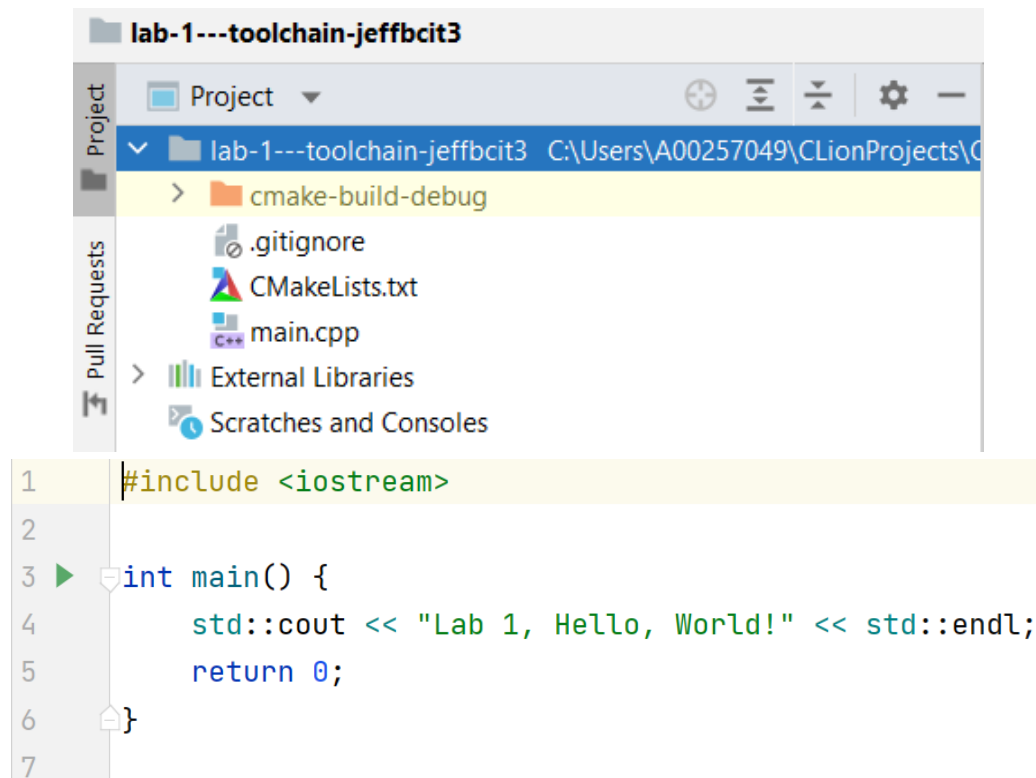


## Getting the Lab 1 code:

8. Go to this link: <https://classroom.github.com/a/YLH3Sjn2> and **Accept this assignment**:
9. Follow the steps similar to points 2 - 7 to **clone the lab 1 repo**.



10. Examine the project files that are created. Note there is a file called `main.cpp`. It contains the main method which prints a message.



11. CLion uses **cmake**, which is a build tool similar to **ant**. Recall that ant uses a file called build.xml. cmake uses a file called CMakeLists.txt. Take a close look at the file. Can you guess what each line represents?
  - a. There is a very good CLion cmake tutorial here:  
<https://www.jetbrains.com/help/clion/quick-cmake-tutorial.html>
12. Ensure the following line is in your CMakeLists.txt in CLion, this adds helpful warning messages for you:

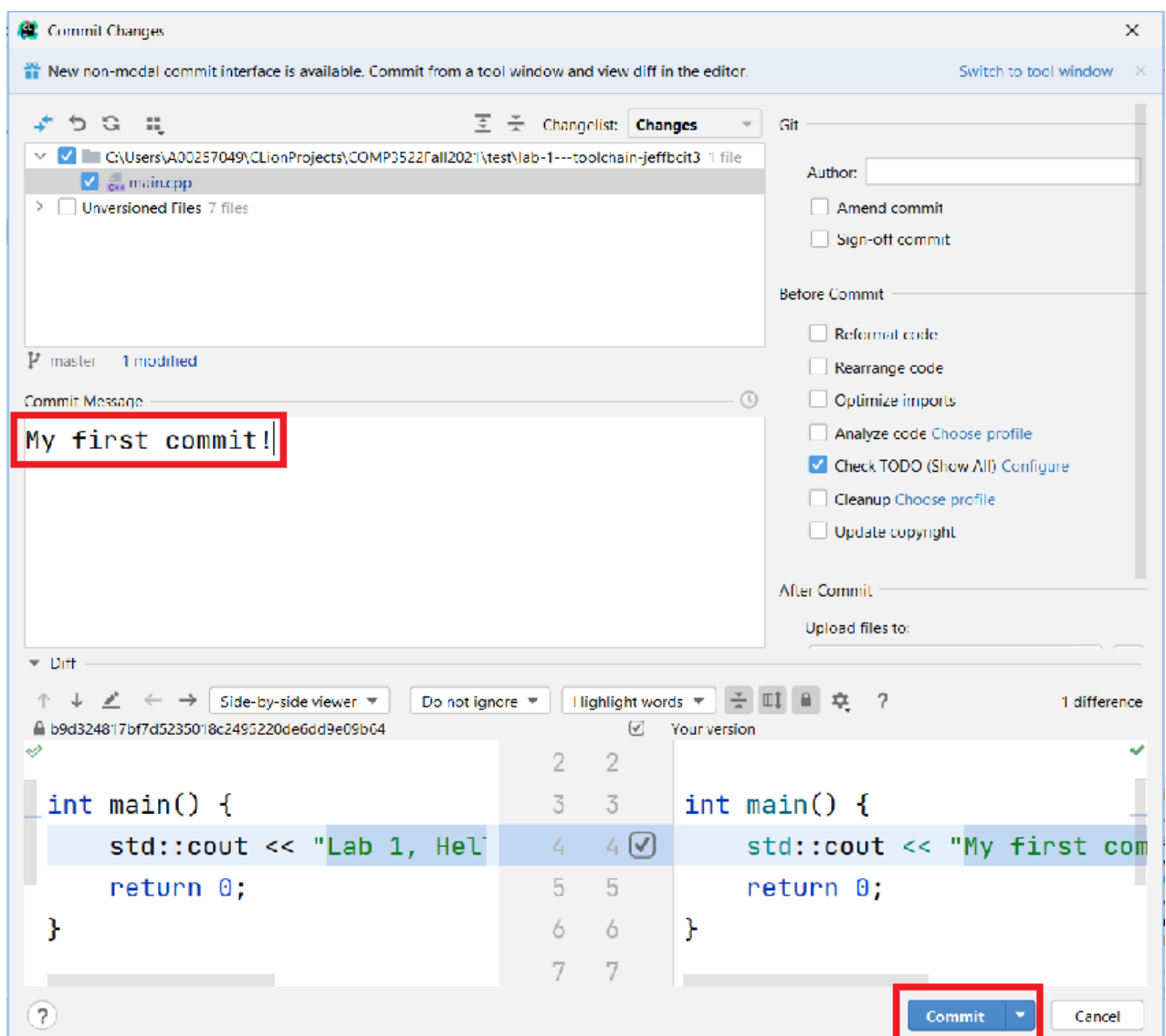
```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Wextra -pedantic")
```

## Commit and Push

13. Modify the "Lab 1, Hello, World" output printed to the screen, and commit your changes.
  - a. Reminder, **committing** saves changes to your local machine. **Pushing** saves changes to the online repo. Make sure to commit and push when you're making large changes and making your final submission to be marked.
14. Select **Git** from the top menu and select **Commit** to commit your changes

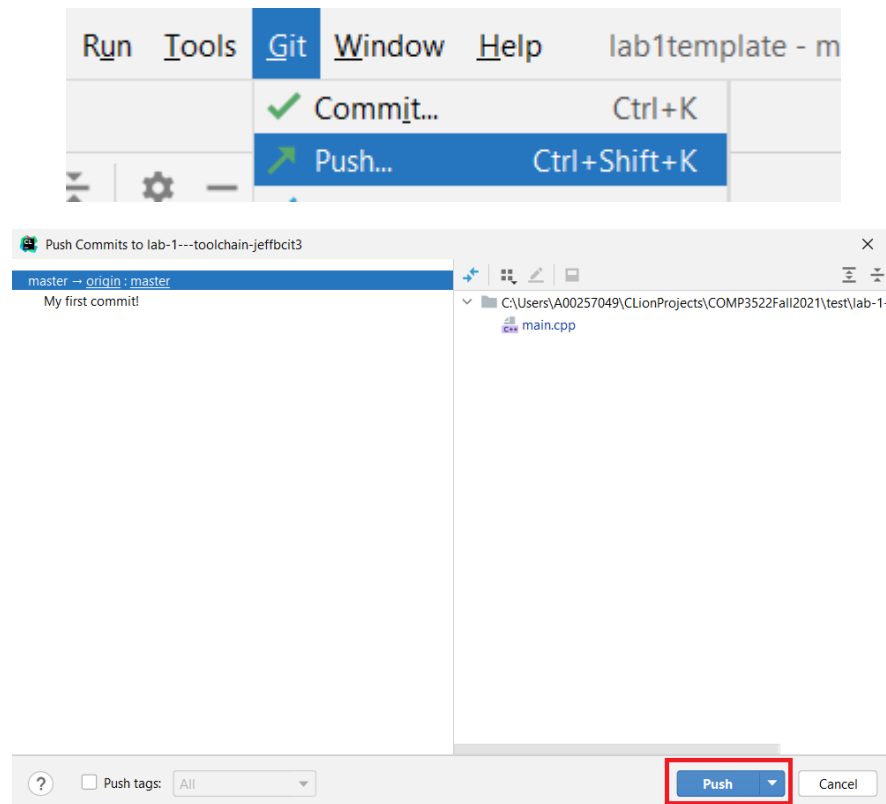


15. Enter a **commit message**, then select **Commit** at the bottom right of the popup to finalize the commit.



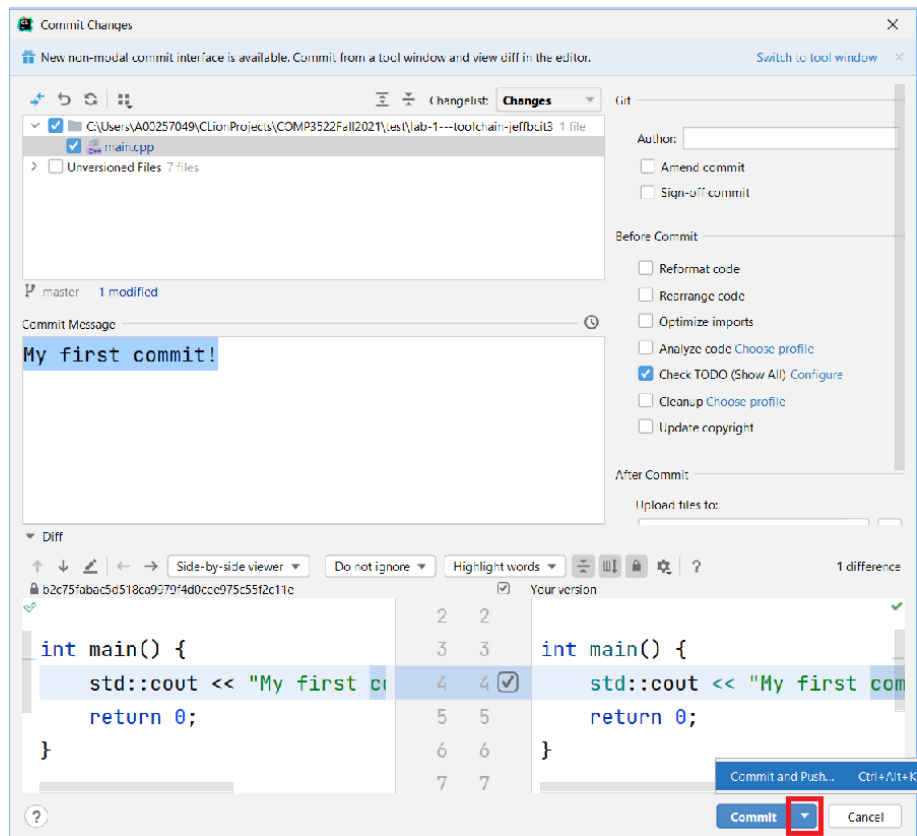
16. Now that the code is committed to your local machine, push the committed code to your online

github repo. Select **Git** and **Push** from the top menu. Then select **Push** in the popup that appears



17. A reminder to commit and push your code regularly.

- a. Protip – you can commit and push at the same time by entering the **commit menu**, then selecting the **down arrow** and selecting **Commit and Push**



# Let's start programming!

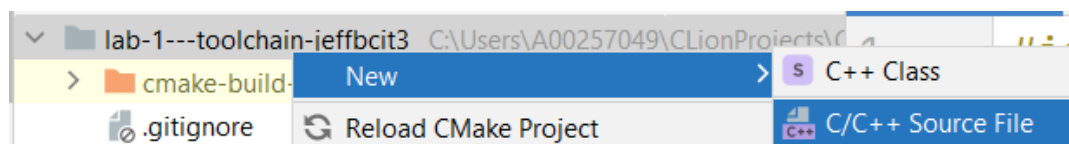
18. Fill out your **name** and **student number** at the top of **main.cpp**
19. Implement a function called gcd. The gcd function accepts two positive integers and returns the greatest common denominator of the two integers. Your function prototype and comment should look like this:

gcd.hpp

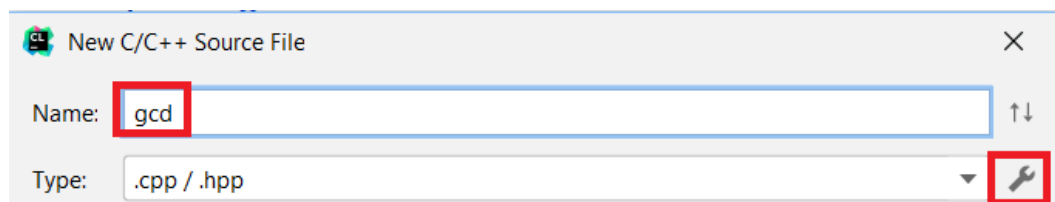
```
// Greatest Common Denominator
// PRE: a is a positive integer
// PRE: b is a positive integer
// POST: a and b are unchanged
// RETURN: the greatest common denominator of
//         a and b.
int gcd(const int a, const int b);
```

20. Ensure you create a .hpp file, not .h. Clion creates .h files by default. The following steps will set this up.

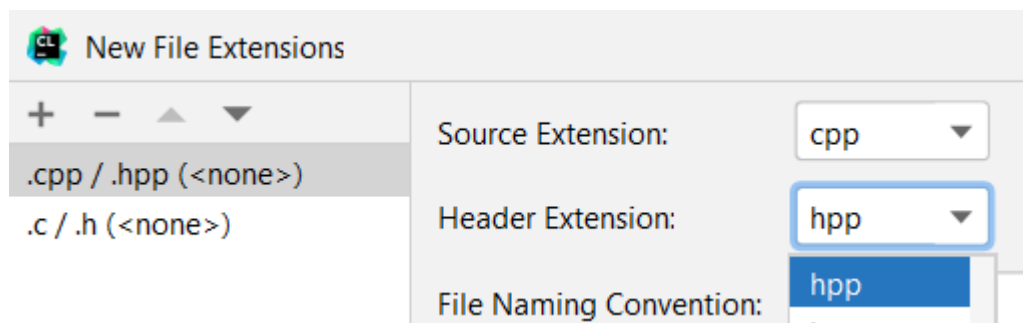
- a. Right click your **lab1 folder** and select **New -> C/C++ Source File**



- b. Enter **gcd** as the name of the file. Press the **wrench icon** in the bottom right

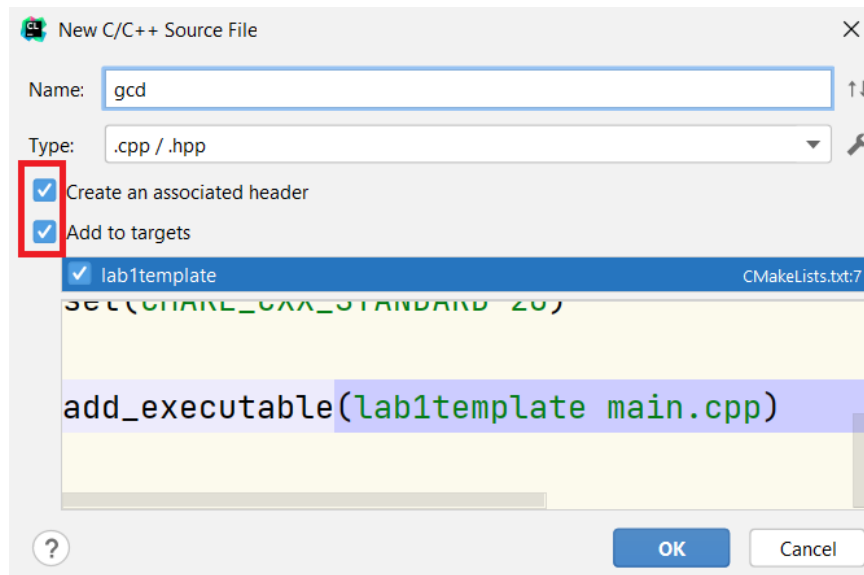


- c. Ensure **source extension is cpp** and **header extension is hpp**. Select **OK** to confirm your changes



- d. Ensure both **Create an associated header** and **Add to targets** checkbox is selected. Then select **OK** to finish creating the gcd files.
  - i. **Create an associated header** – automatically creates a .hpp header file along with the .cpp source file
  - ii. **Add to targets** - adds the new .hpp and .cpp file to the **CMakeLists.txt** file.

This is required for the new files to be compiled as part of your project



21. Write the gcd function **prototype** in the **gcd.hpp** header file
22. Write the gcd function's **complete implementation** in the **gcd.cpp** file
23. Write code in **main.cpp** to call your gcd function to show it works. Make sure to **import the gcd function** into main.cpp by adding **#include gcd.hpp** at the top of main.cpp
24. Ensure you commit and push your new function. Prove it works by invoking it in the main function.
25. Do NOT submit this Lab via D2L/The Learning Hub, submit by **pushing it to your GitHub repo via CLion**
26. That's it, you're finished lab 1!

## 2 Grading

This lab and all future labs will be marked out of 10. For full marks this week, you must:

1. (2 points) Install or have already installed current versions of CLion, g++, and git
2. (3 point) Successfully configure CLion to work with g++, GitHub, and GitHub Classroom
3. (2 points) Commit and push to GitHub after each non-trivial change to your code
4. (3 points) Implement and test the gcd function described above (remember to create a separate header file for the function prototype).

### Test cases:

Try out these two numbers, along with your own custom test cases:

Find the GCD of: **5764375, 46875**

### Troubleshooting:

If you're a mac user and getting an error, when running any code, similar to the following:

The C compiler

"/usr/bin/cc"

is not able to compile a simple test program.

It fails with the following output:

Change Dir: /Users/a1/Cluster\_analys/cmake-build-debug/CMakeFiles/CMakeTmp

It may be because you haven't installed the xcode developer tools before installing brew and gcc. In that case:

- Please open the Terminal, and run the following command: `xcode-select --install`
- After that in CLion please select `Tools | CMake | Reset Cache and Reload Project`

credit:

<https://intellij-support.jetbrains.com/hc/en-us/community/posts/360010551220-Can-t-compile-simple-test-program-on-Mac>

Credit to Jayden, if you need access to the g++ executable and can't find the location, here's how to do it:

"I believe I've found the location of the actual g++ executable file

1. Go to Macintosh HD directory
2. Press cmd + shift + .(dot) (simultaneously) and it will reveal hidden files + folders
3. Go into /usr/bin, where you should find both the gcc & g++ executable files"