

CS3A-classroom / lab0_writeup

[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[master](#) ▾

...

lab0_writeup / windows.md



barkeshli two missing images

[History](#)

2 contributors

[Raw](#)[Blame](#)

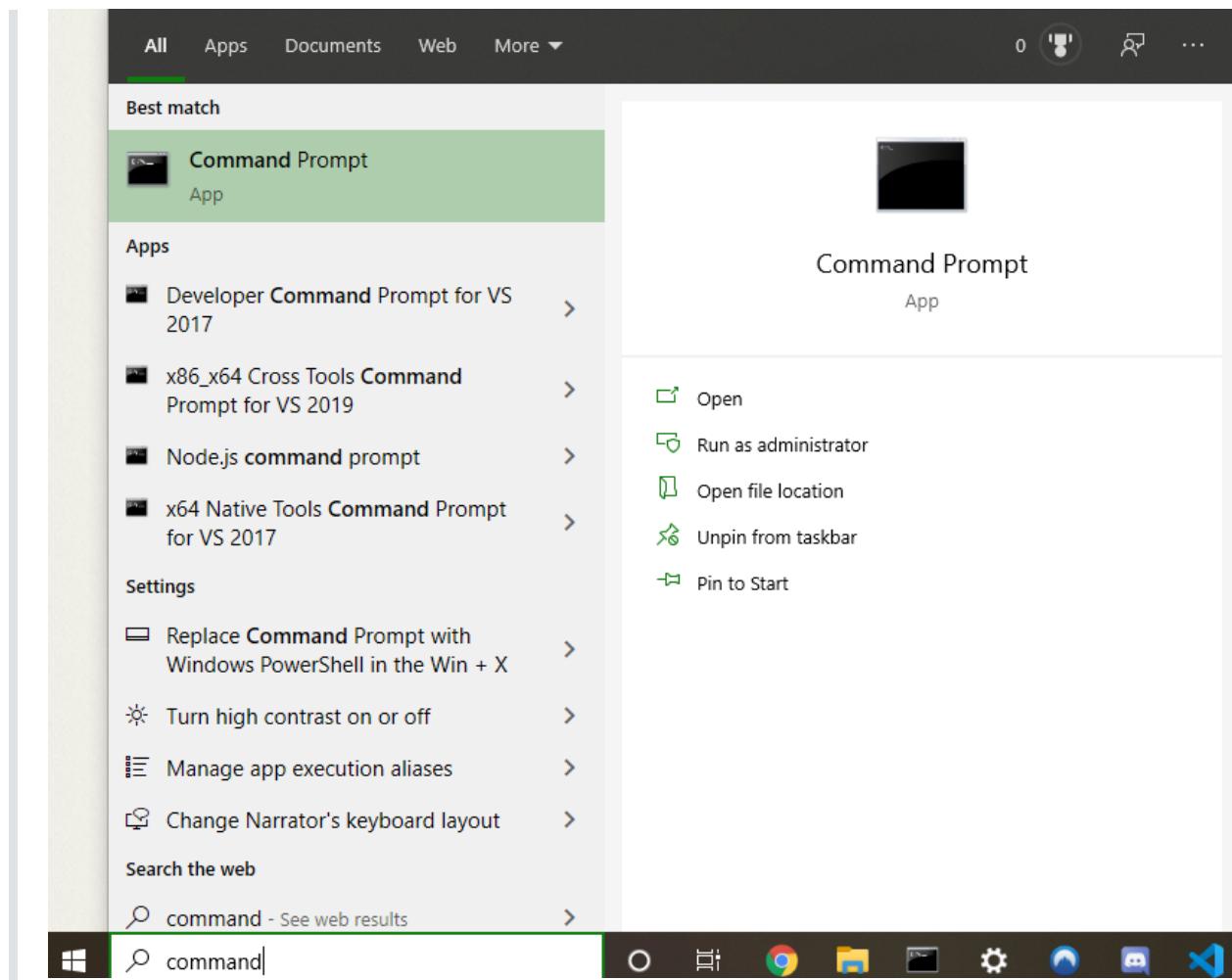
1210 lines (655 sloc) | 38.8 KB

Windows Instructions

- [opening commandline](#)
- [Installing git](#)
- [Installing cmake](#)
- [Installing MinGW](#)
- [Accepting the assignment](#)
- [Project organization](#)
- [Quick edit, status , add , commit , & push](#)

- **Build and run walkthrough**
- **Getting started with the project**
- **Writing tests**
- **Completing the project**

■ Opening the terminal ■

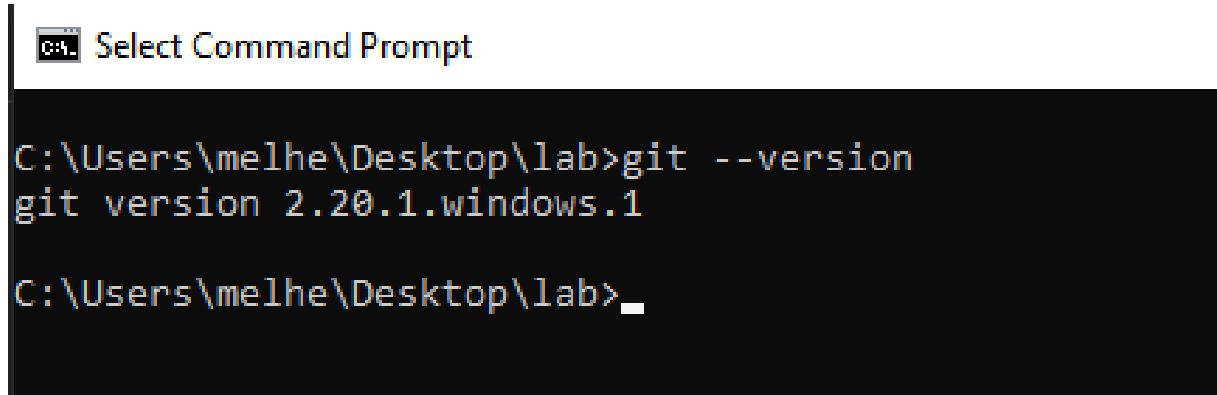


■ Installing git ■

is git installed?

Let's check to see if `git` is installed on your system: type `git --version` at the commandline.

If you do not get a response similar to this, then you do not have `git` on your system and you have to install it.



The screenshot shows a Windows Command Prompt window with a title bar labeled "Select Command Prompt". The window contains the following text:

```
C:\Users\melhe\Desktop\lab>git --version
git version 2.20.1.windows.1

C:\Users\melhe\Desktop\lab>
```

Download git

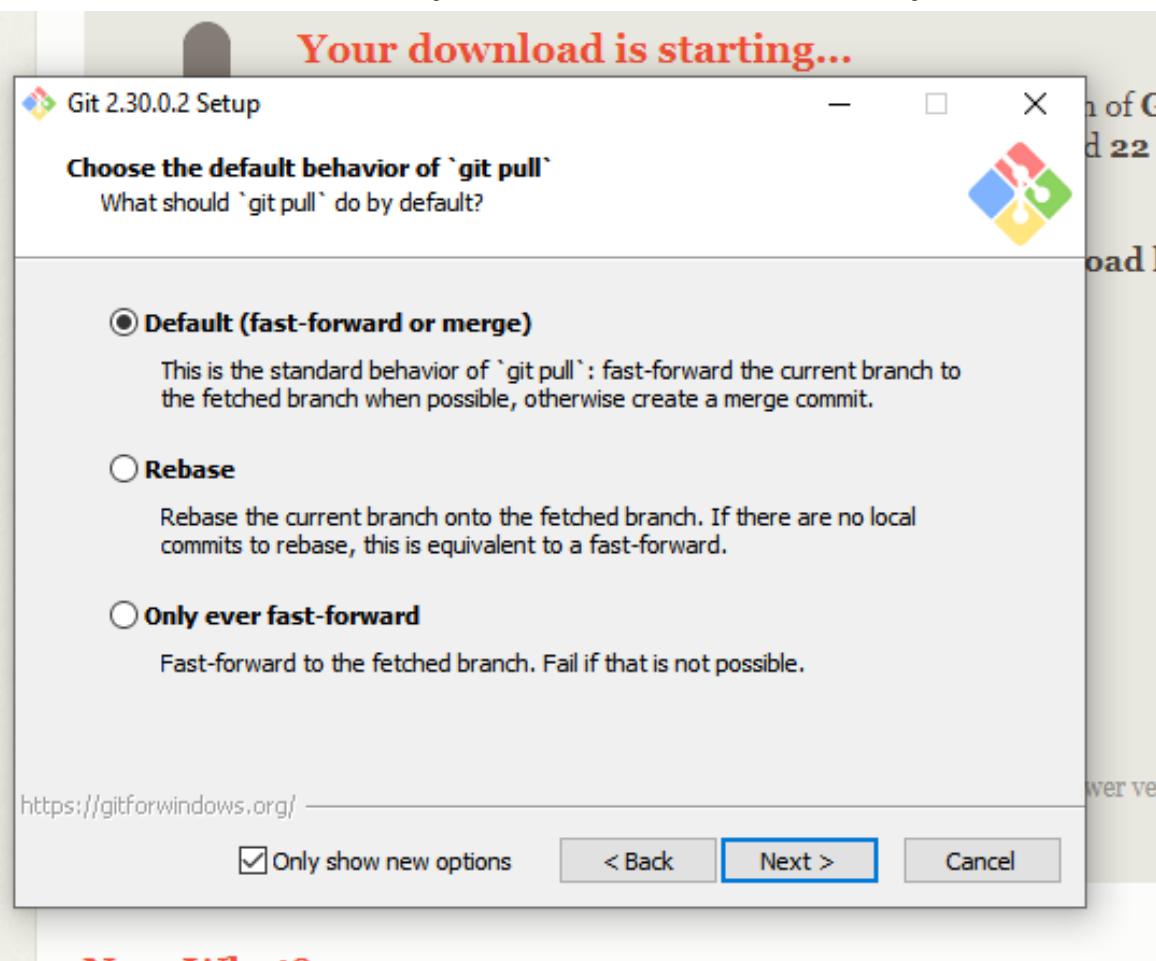
Download git from [here](#). You should get an exe similar to this: Git-2.30.0.2-64-bit.exe

The screenshot shows the official Git website's download page. At the top left is the Git logo with the tagline "distributed-is-the-new-centralized". A search bar is at the top right. On the left sidebar, there are links for About, Documentation, Downloads (selected), GUI Clients, Logos, and Community. Below the sidebar is a box containing information about the "Pro Git book" by Scott Chacon and Ben Straub, available online for free and on Amazon. The main content area has a large downward arrow icon and the heading "Downloading Git". It says "Your download is starting..." and provides a link to download manually if the download hasn't started. It lists "Other Git for Windows downloads" including Git for Windows Setup, 32-bit Git for Windows Setup, 64-bit Git for Windows Setup, Git for Windows Portable ("thumbdrive edition"), 32-bit Git for Windows Portable, and 64-bit Git for Windows Portable. It also notes the current source code release is version 2.30.0. Below this is a section titled "Now What?" with three options: "Read the Book", "Download a GUI", and "Get Involved".

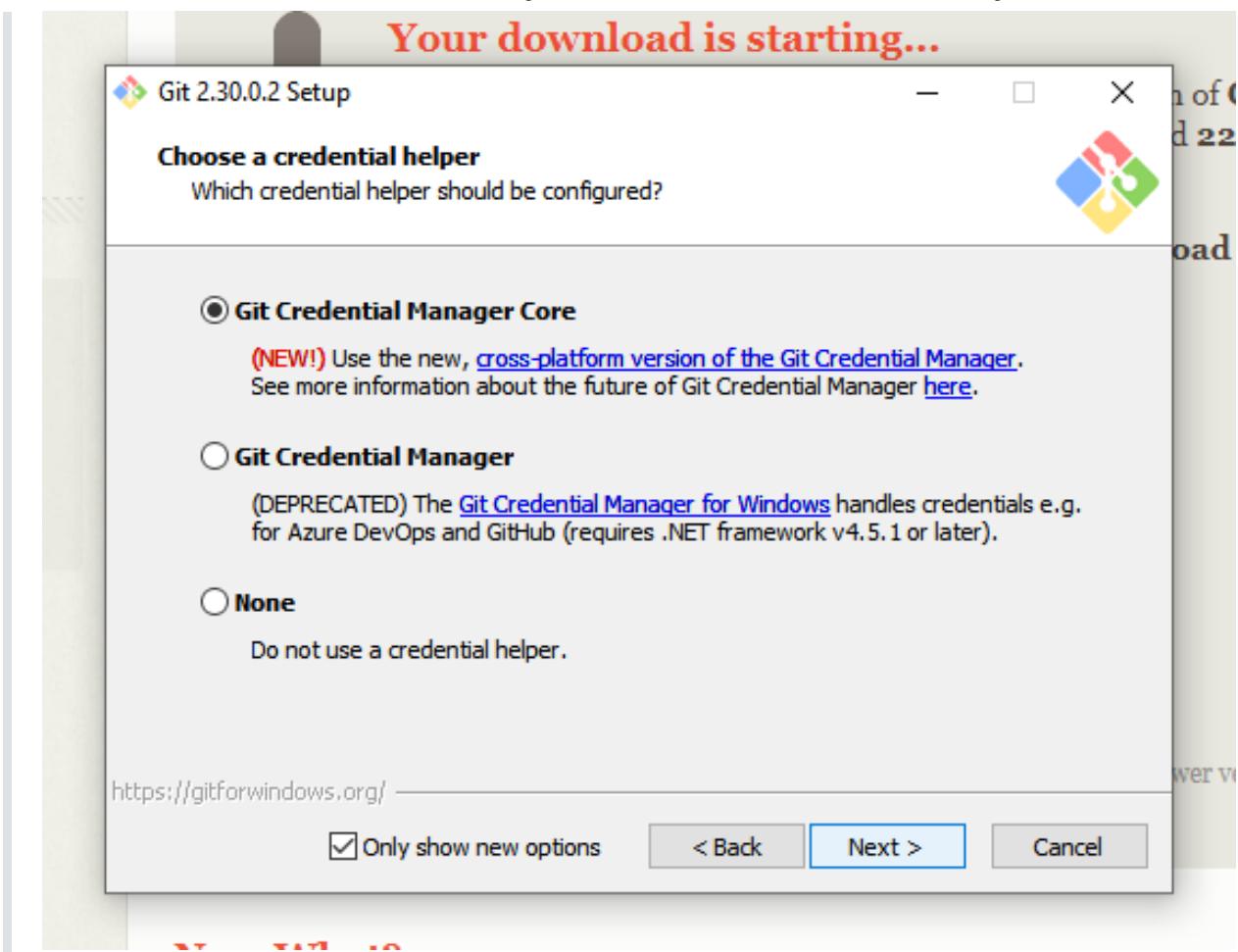
Install git

Open/run the executable, and follow the steps to install.

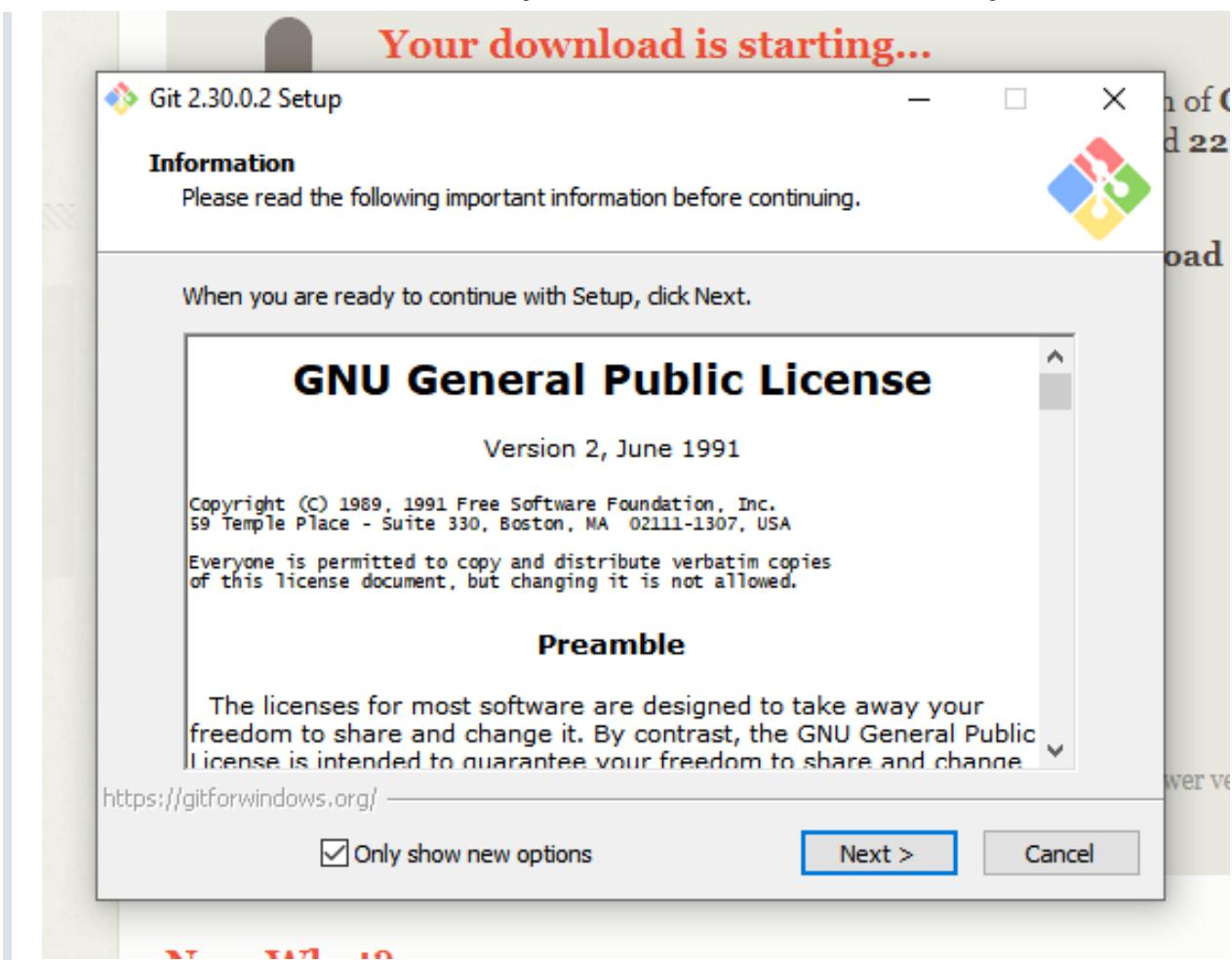
Choose Default.



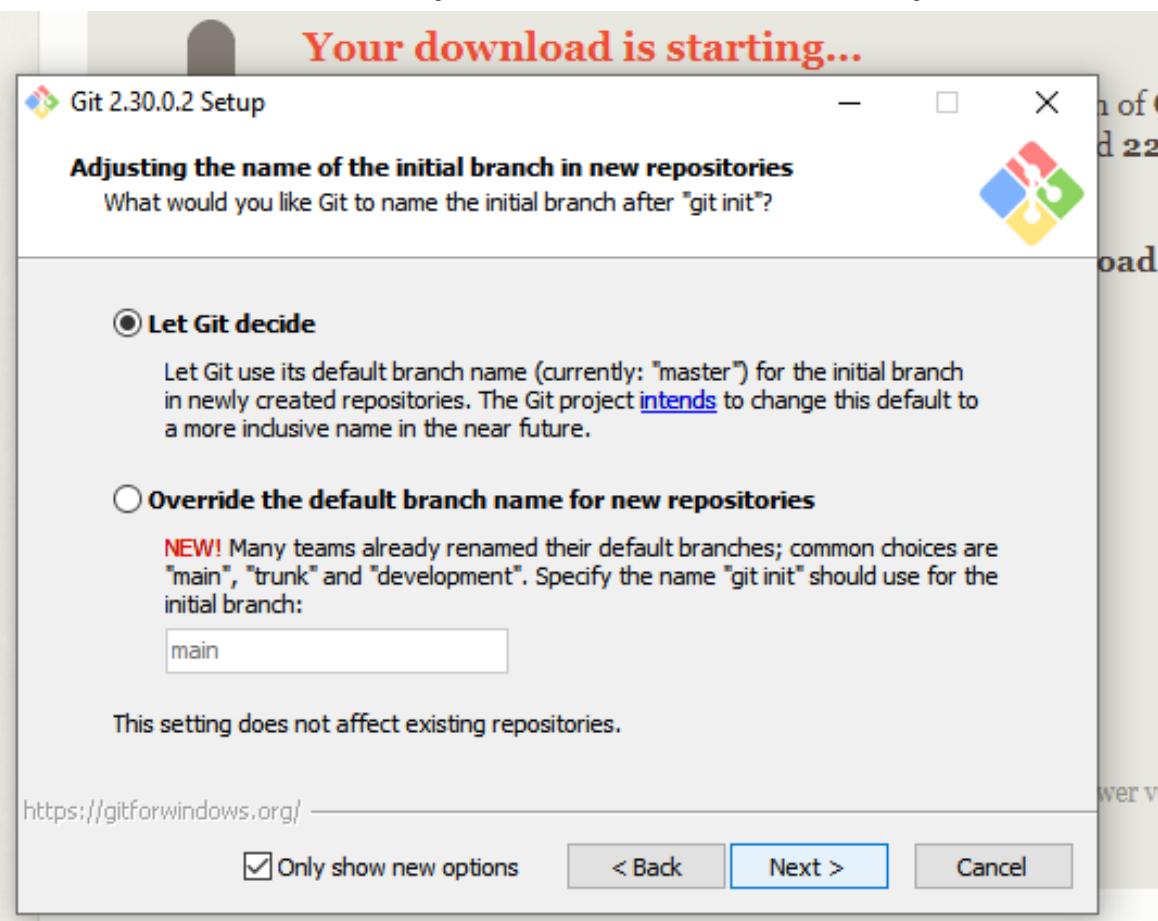
Choose Git Credential Manager Core.



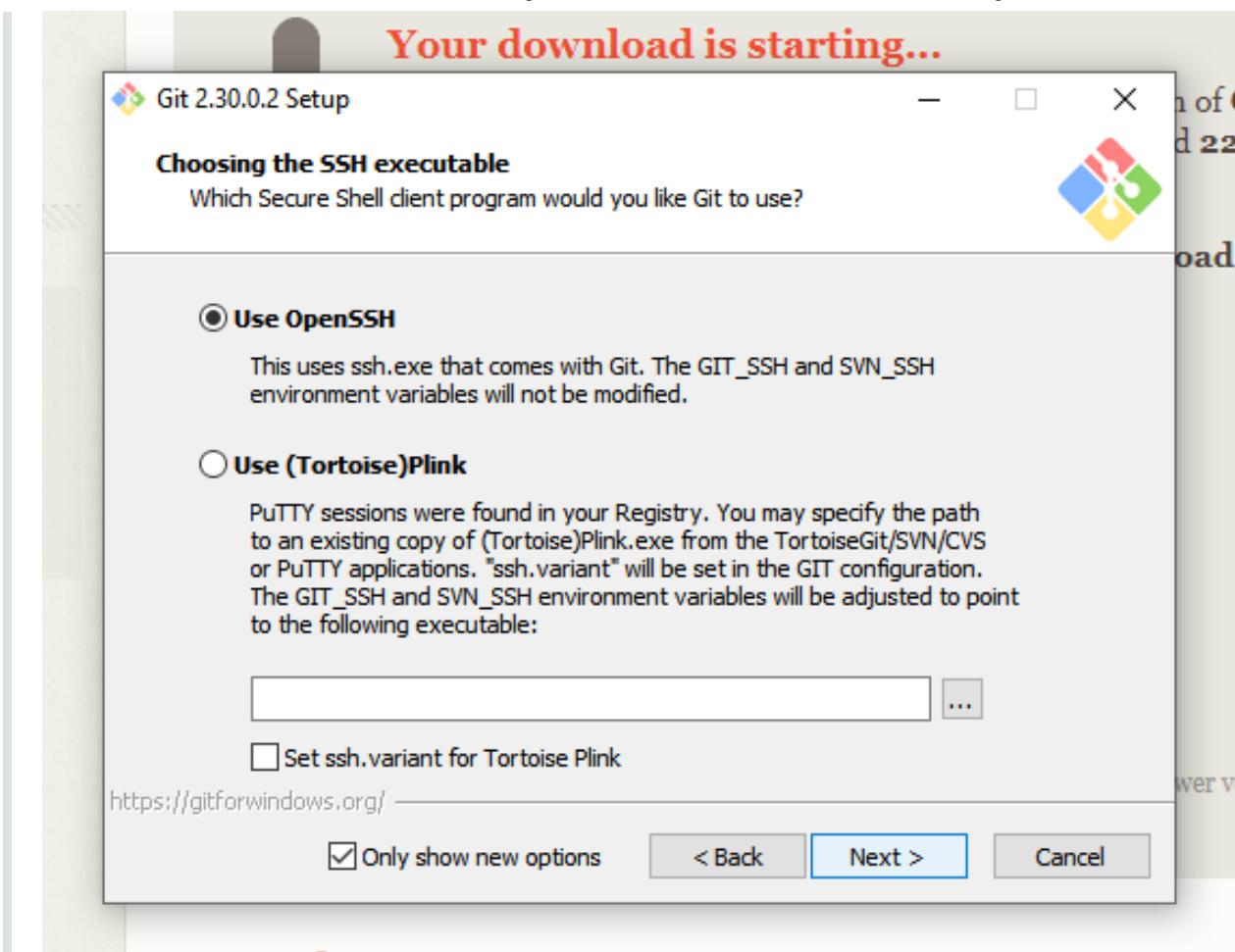
Select Only show new options if not selected, and click next.



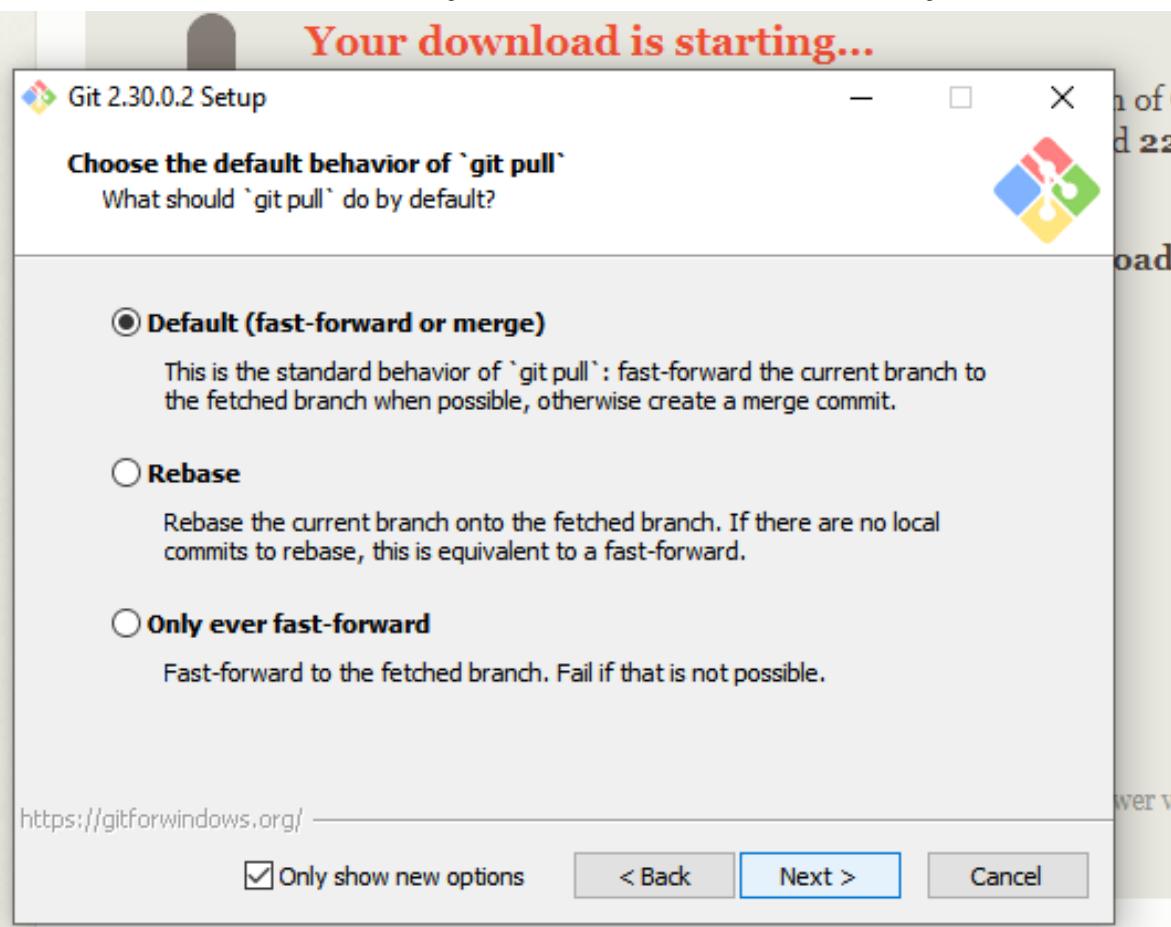
Choose Let git decide.



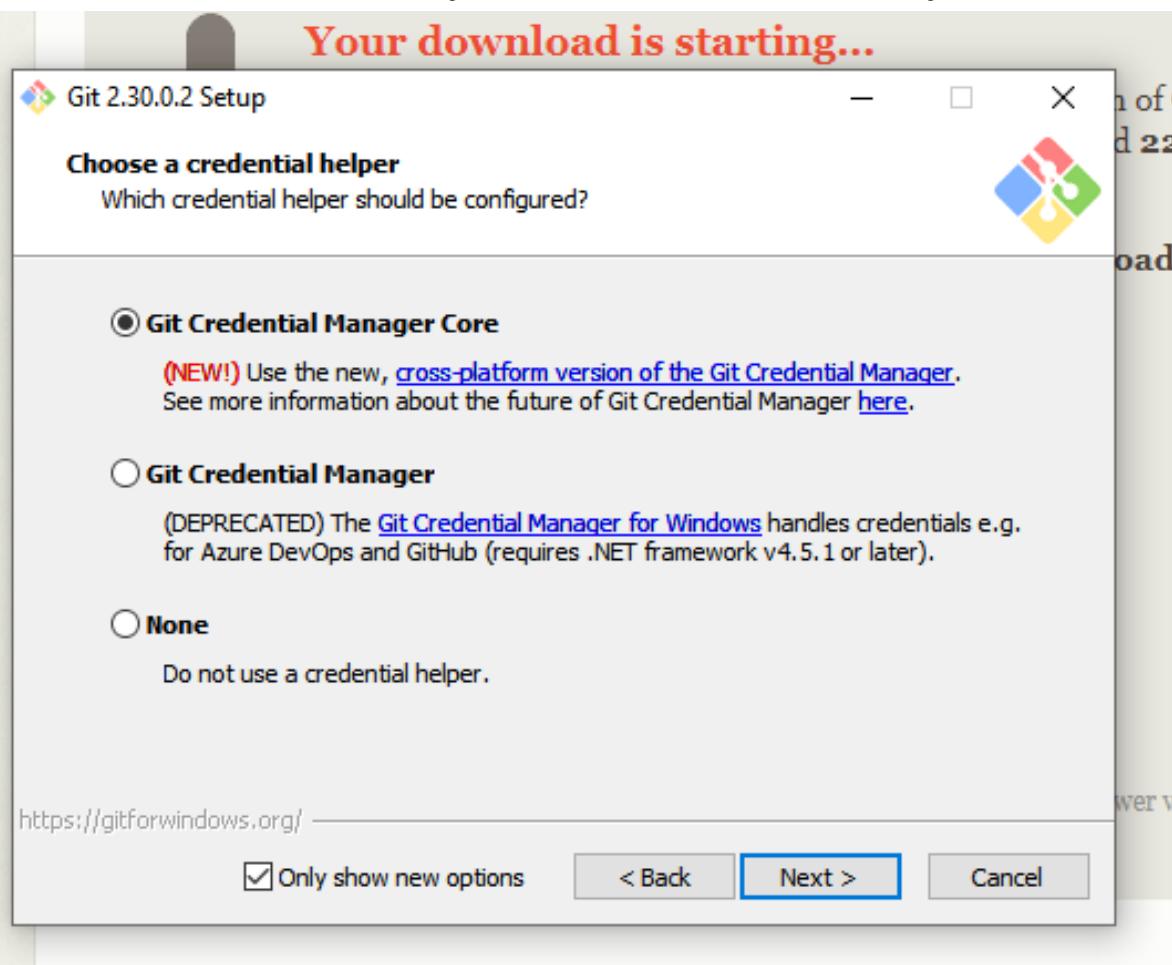
Choose Use OpenSSH.



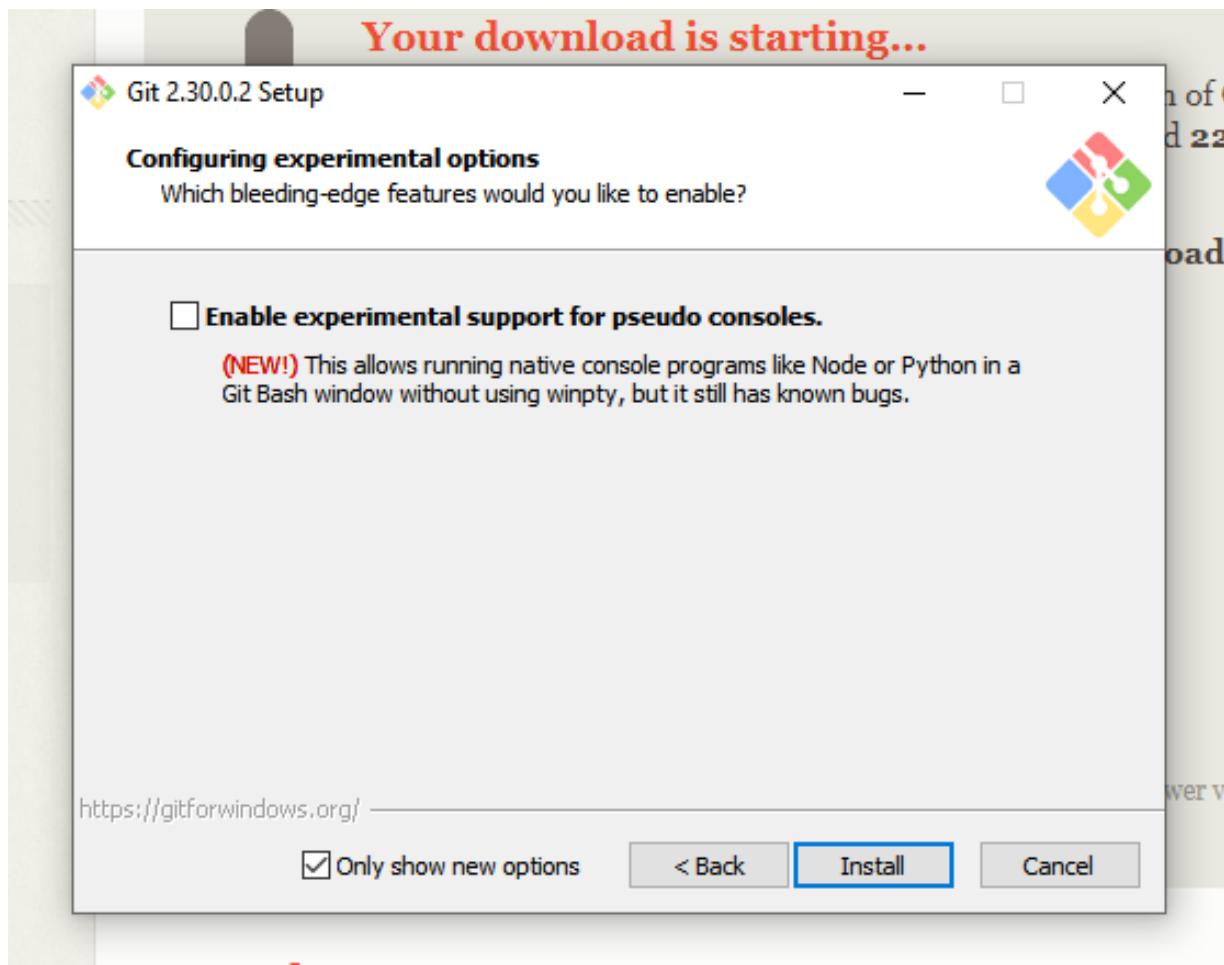
Choose Default (fast-forward or merge).

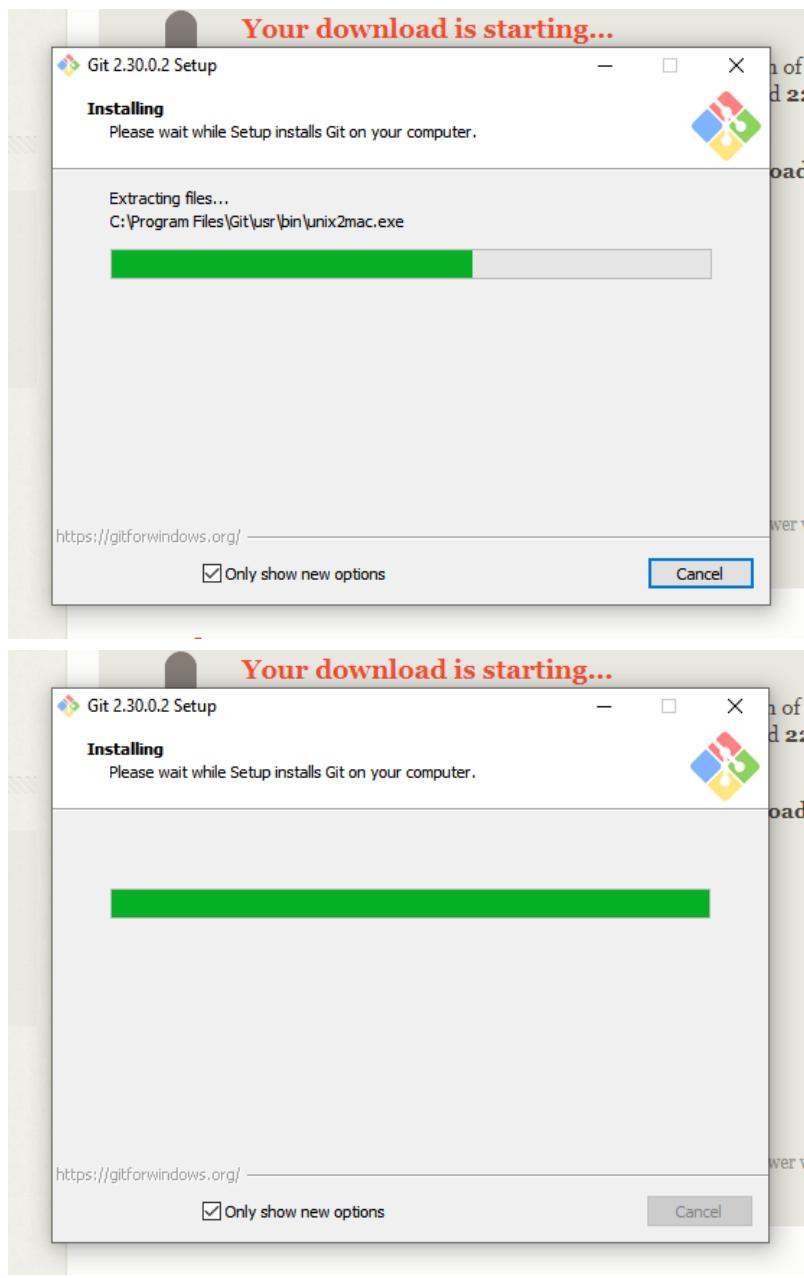


Choose Git Credential Manager Core

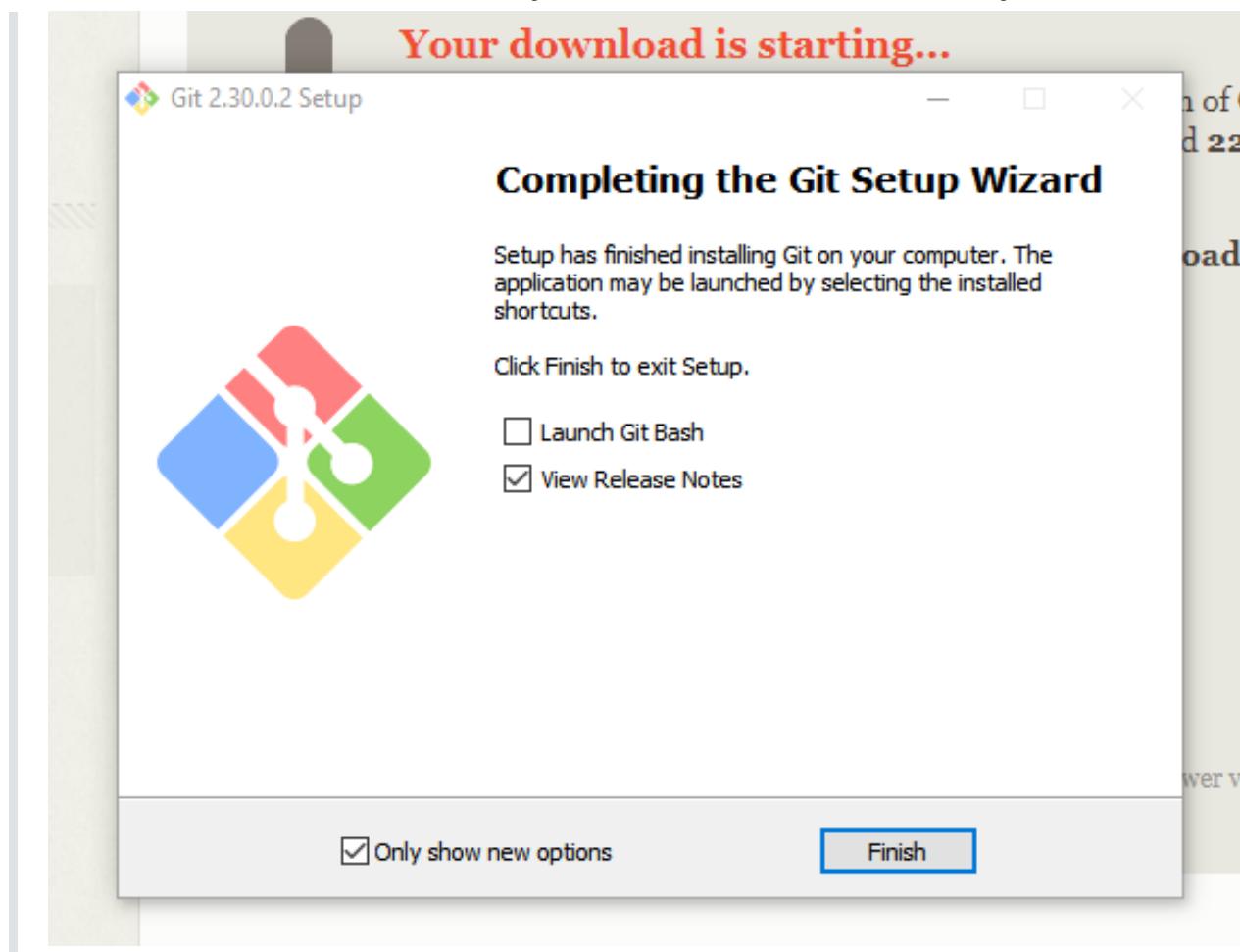


And Install!





When it is done installing, click Finish.

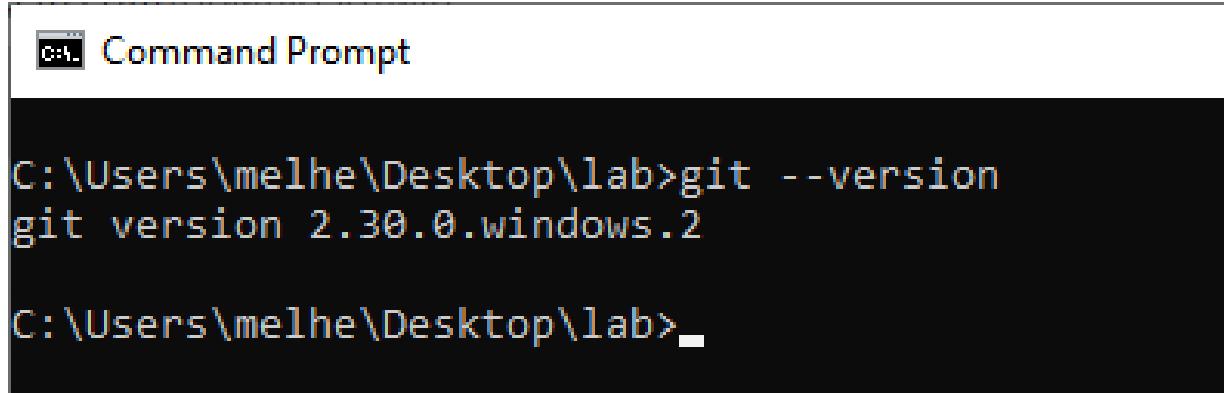


You may view release notes if you want.

A screenshot of the "Git for Windows v2.30.0(2) Release Notes" page. The header includes a navigation menu with links like "HOMEPAGE", "RELEASES", "CONTRIBUTE", "ISSUES", and "QUESTIONS", and a logo. The main content area has a dark blue background. It starts with an "Introduction" section, followed by a note about specific issues for the Windows release. Below that is a "Known issues" section with a note about reporting bugs. There are expandable sections for "Licenses" and "Changes in v2.30.0(2) since v2.30.0 (December 28th 2020)". The "Changes" section is currently expanded, showing a note about Git LFS v2.13.2 addressing CVE-2021-21237. At the bottom, there's a "New Features" section with a note about Git Credential Manager Core v2.0.318.44100 and Git LFS v2.13.2.

Check the version of the git again:

To make sure `git` is installed correctly, run `git --version` again:



```
Command Prompt

C:\Users\melhe\Desktop\lab>git --version
git version 2.30.0.windows.2

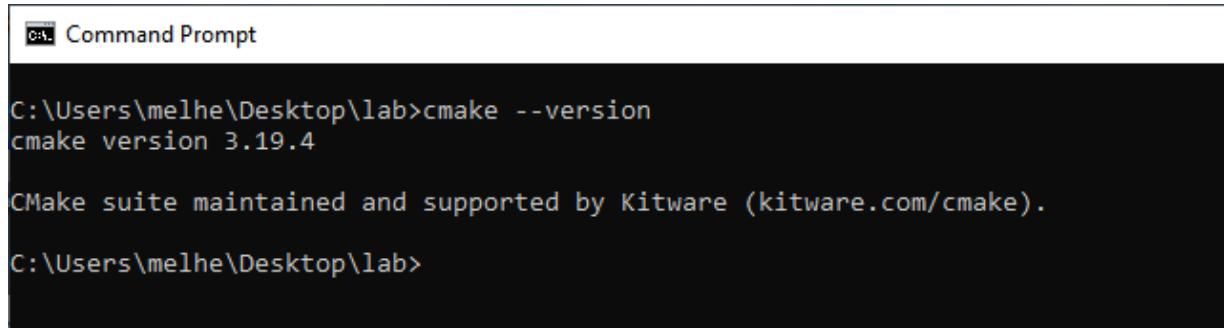
C:\Users\melhe\Desktop\lab>
```

■ Installing cmake ■

is `cmake` installed?

Let's check to see if `cmake` is installed on your system: type `cmake --version` at the commandline.

If you do not get a response similar to this, then you do not have `cmake` on your system and you have to install it.



```
Command Prompt

C:\Users\melhe\Desktop\lab>cmake --version
cmake version 3.19.4

CMake suite maintained and supported by Kitware (kitware.com/cmake).

C:\Users\melhe\Desktop\lab>
```

Download cmake

Download cmake from [here](#). Choose the Windows win64-x64 Installer. You should get an msi with a name similar to this: `cmake-3.19.4-win64-x64.msi`

The screenshot shows the CMake download page. At the top, there's a navigation bar with links for About, Resources, Developer Resources, Download, and a search icon. A banner on the right says "KITWARE IS HIRING". Below the navigation, there's a note about Linux-XO_04 prefixes. The main content is divided into two sections: "Source distributions:" and "Binary distributions:". Each section contains a table with "Platform" and "Files" columns.

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.19.4.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.19.4.zip

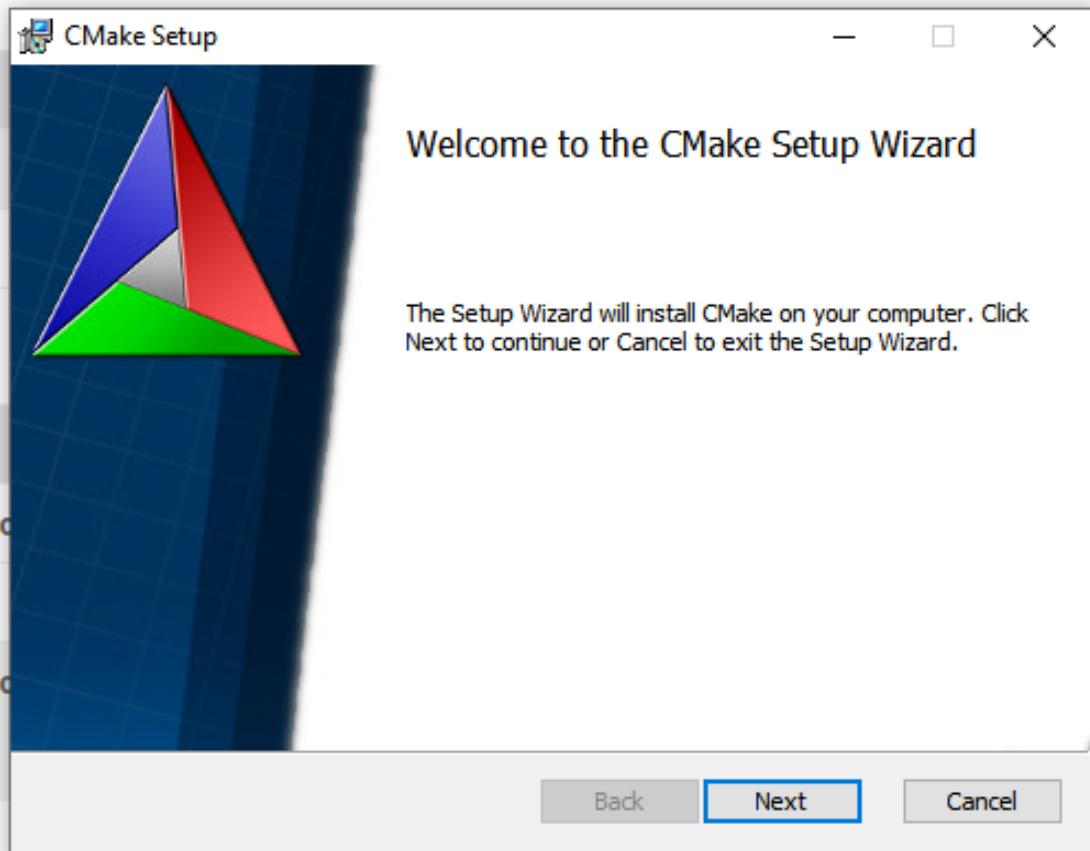
Platform	Files
Windows win64-x64 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.19.4-win64-x64.msi
Windows win64-x64 ZIP	cmake-3.19.4-win64-x64.zip
Windows win32-x86 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.19.4-win32-x86.msi
Windows win32-x86 ZIP	cmake-3.19.4-win32-x86.zip
Mac OS X 10.13 or later	cmake-3.19.4-macos-universal.dmg
	cmake-3.19.4-macos-universal.tar.gz
Mac OS X 10.10 or later	cmake-3.19.4-macos10.10-universal.dmg
	cmake-3.19.4-macos10.10-universal.tar.gz
Linux x86_64	cmake-3.19.4-Linux-x86_64.sh
	cmake-3.19.4-Linux-x86_64.tar.gz
Linux aarch64	cmake-3.19.4-Linux-aarch64.sh
	cmake-3.19.4-Linux-aarch64.tar.gz

Below the tables, there's a section for "Download verification:" with a table:

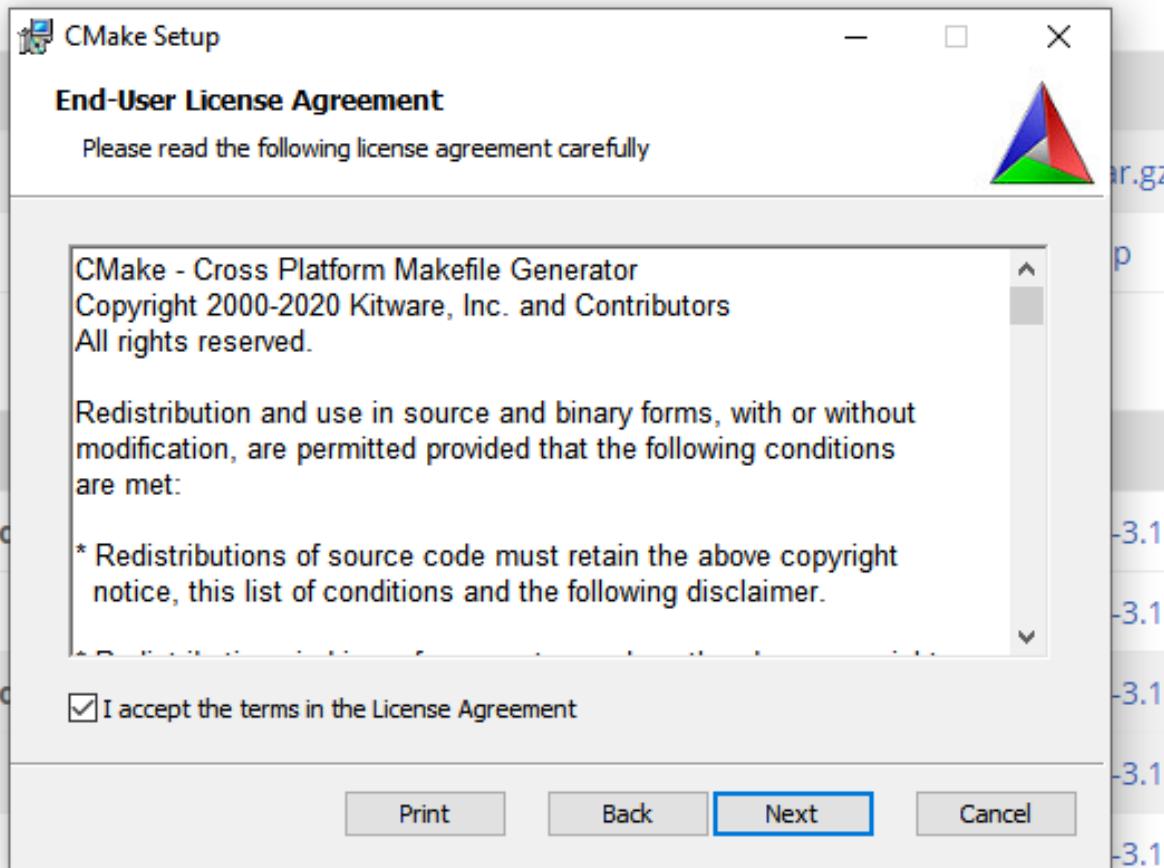
Role	Files
------	-------

Install cmake

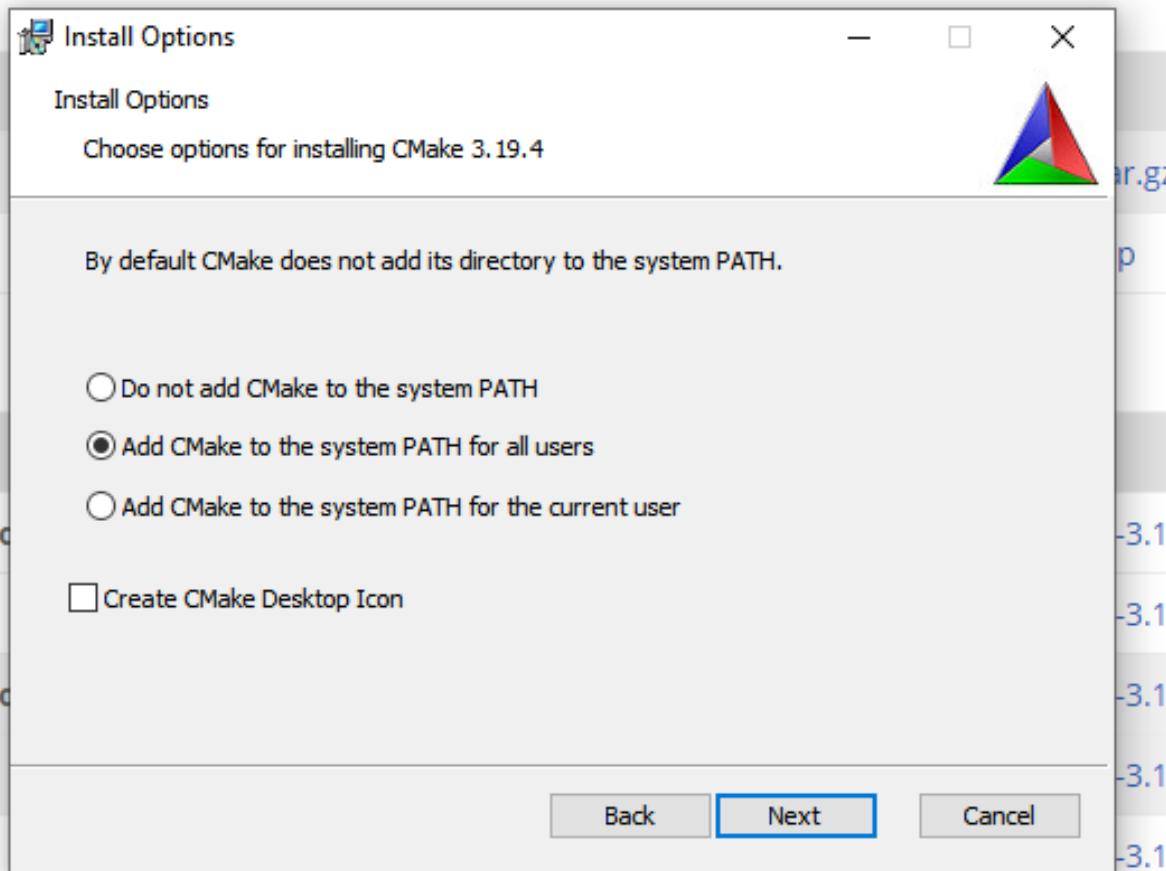
Open/run the executable, and follow the steps to install.



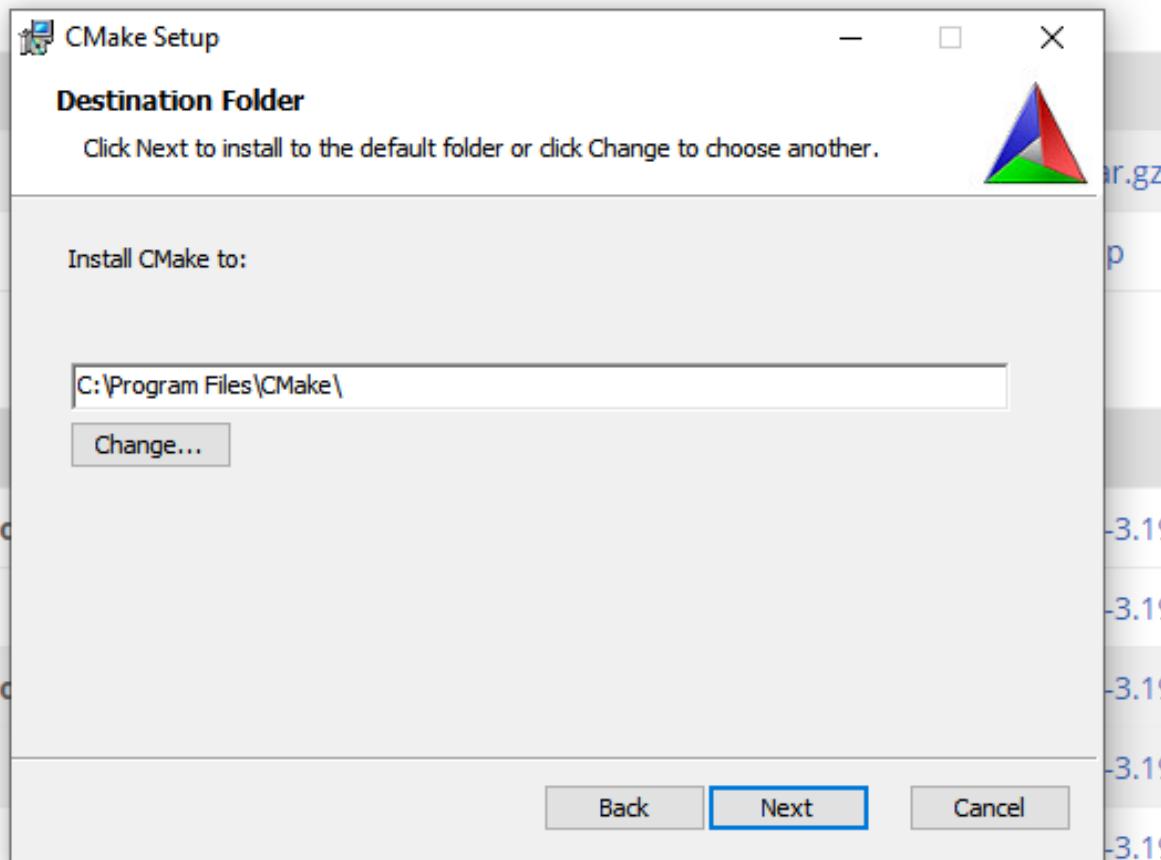
Check I accept the terms in the Liscense Agreement.



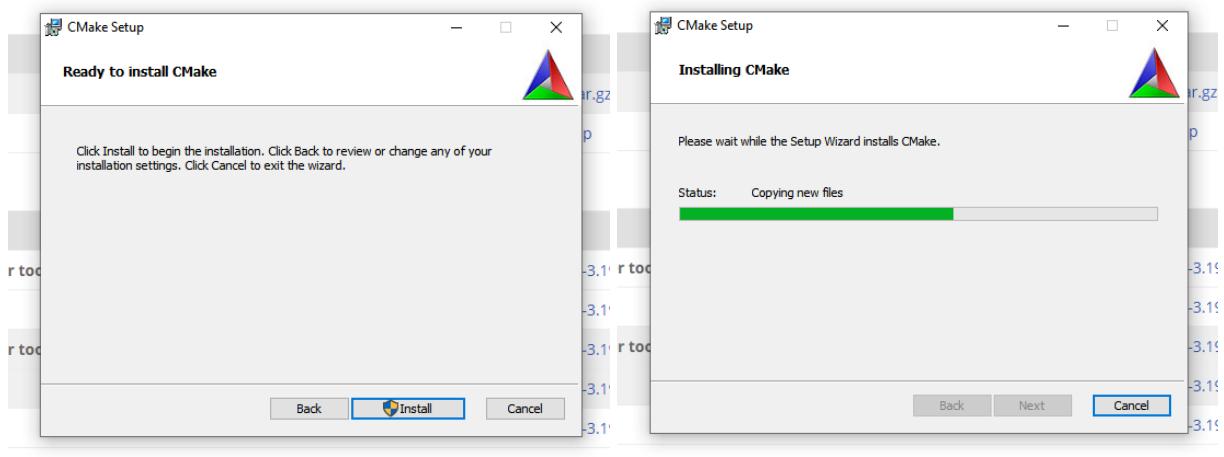
Make sure to select Add CMake to the system PATH for all users. You can create a Desktop icon if you want, but you will not need to use it for this class.



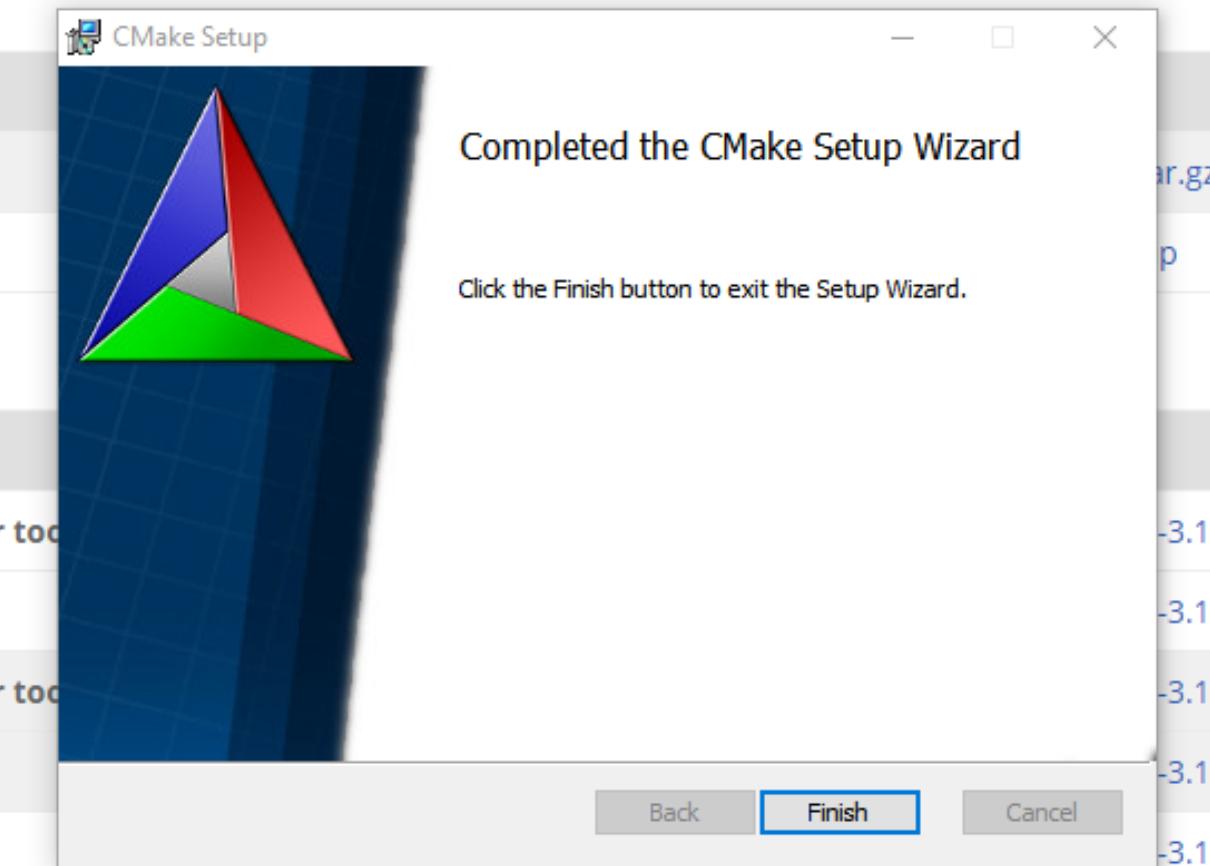
The default install location wshould be C:\Program Files\CMake



Install!



When it is done installing, click Finish.



Check the version of the git again:

To make sure `cmake` is installed correctly, run `cmake --version` again:

```
Command Prompt

C:\Users\melhe\Desktop\lab>cmake --version
cmake version 3.19.4

CMake suite maintained and supported by Kitware (kitware.com/cmake).

C:\Users\melhe\Desktop\lab>
```

■ Install MinGW ■

is MinGW installed?

Let's check to see if `g++` is installed on your system: type `g++ --version` at the commandline.

If you do not get a response similar to this, then you do not have `g++` and/or `MinGW` on your system and you have to install it.

```
C:\ Command Prompt

C:\Users\melhe\Desktop\lab>g++ --version
g++ (MinGW.org GCC Build-2) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

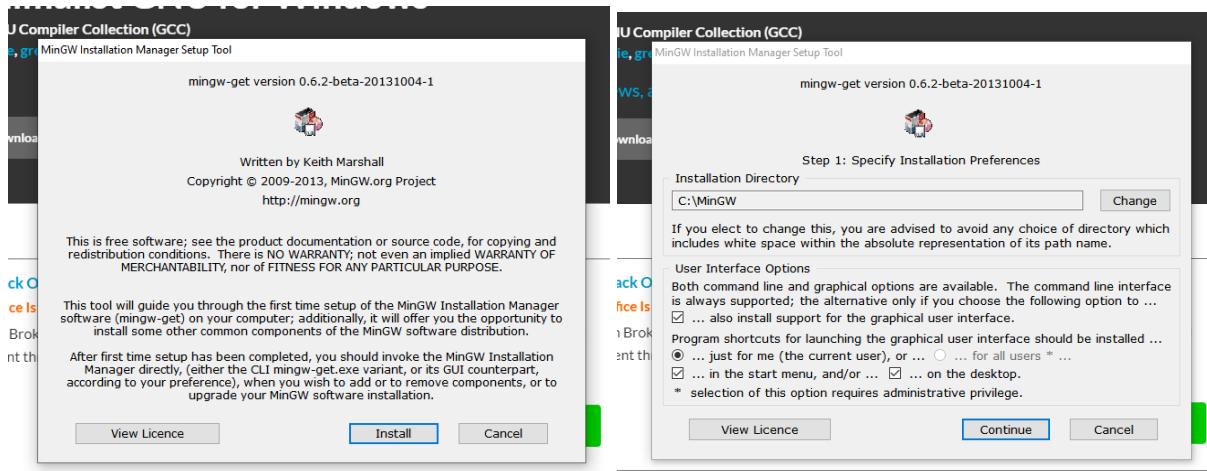
C:\Users\melhe\Desktop\lab>
```

Download MinGW

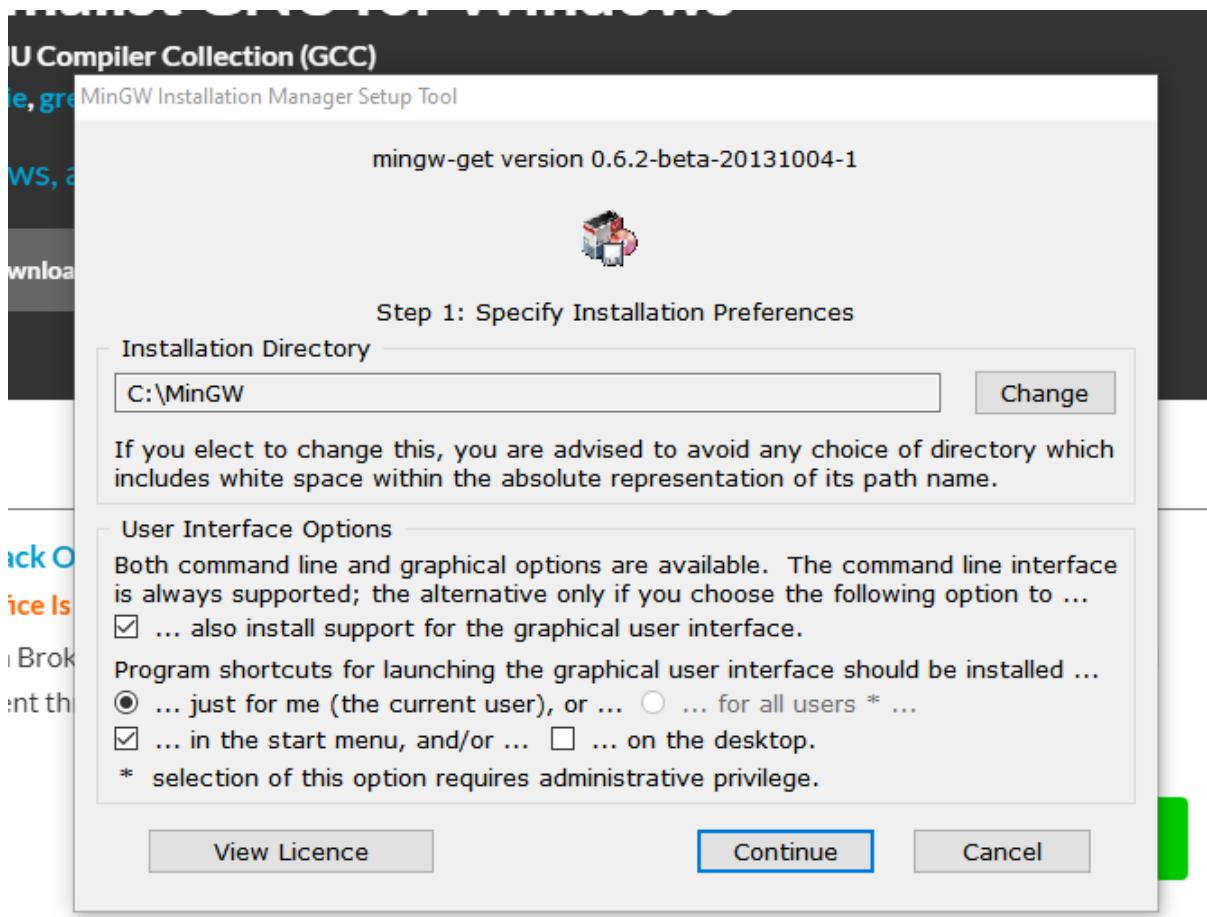
Download cmake from [here](#). You should get an exe named similar to this: `mingw-get-setup (1).exe`

Install MinGW / g++

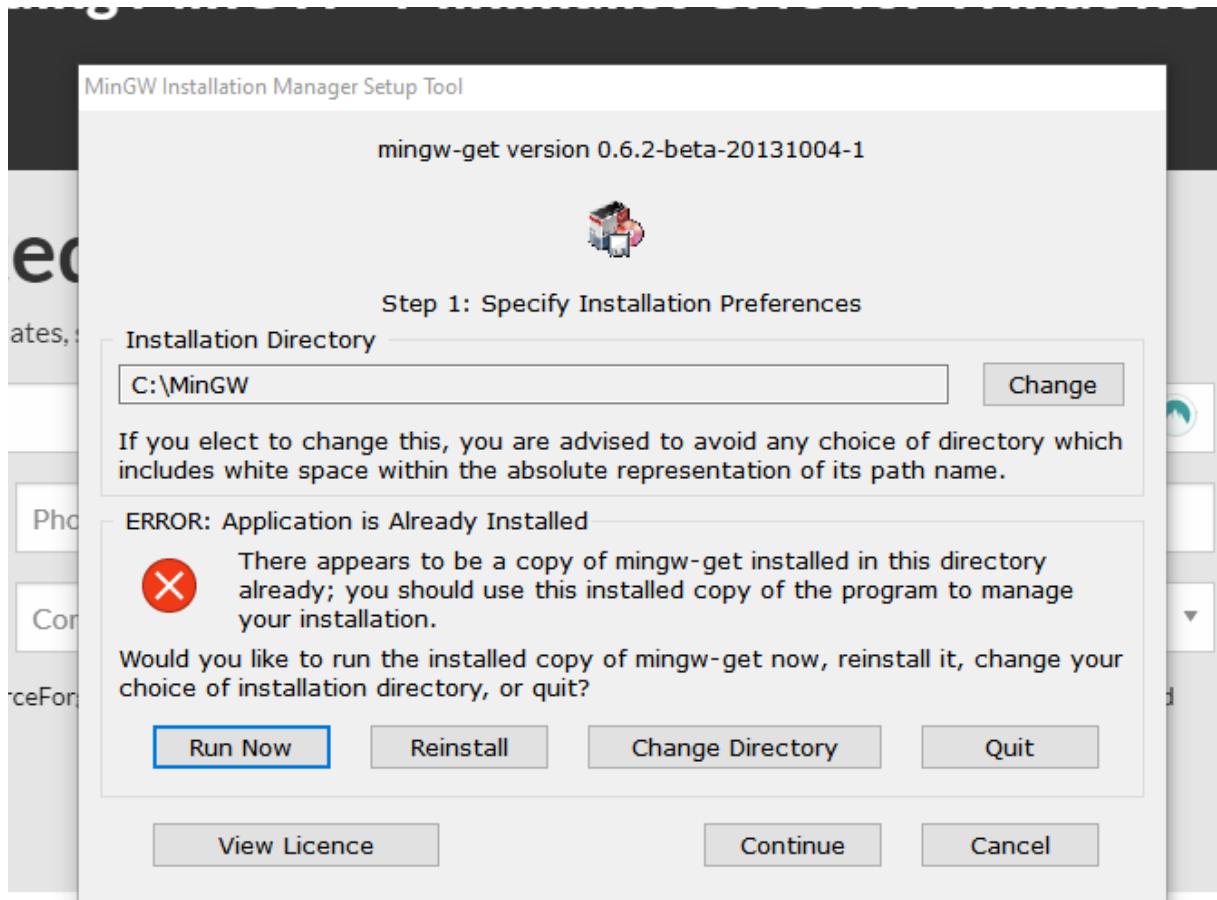
Open/run the executable, and follow the steps to install.



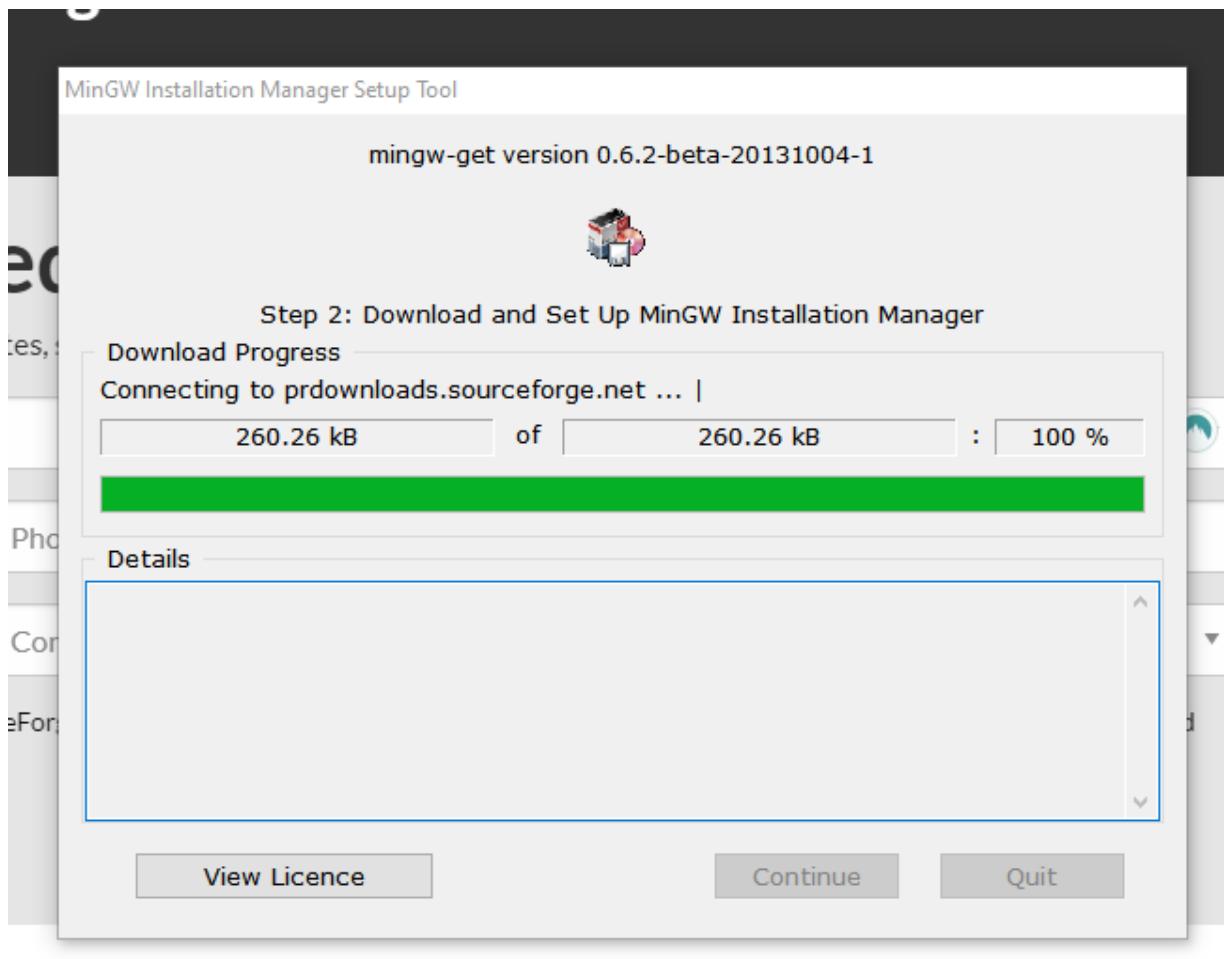
The default Installation location should be C:/MinGW. Once again, you can add a Desktop shortcut, but it will not be neccesary for this class.



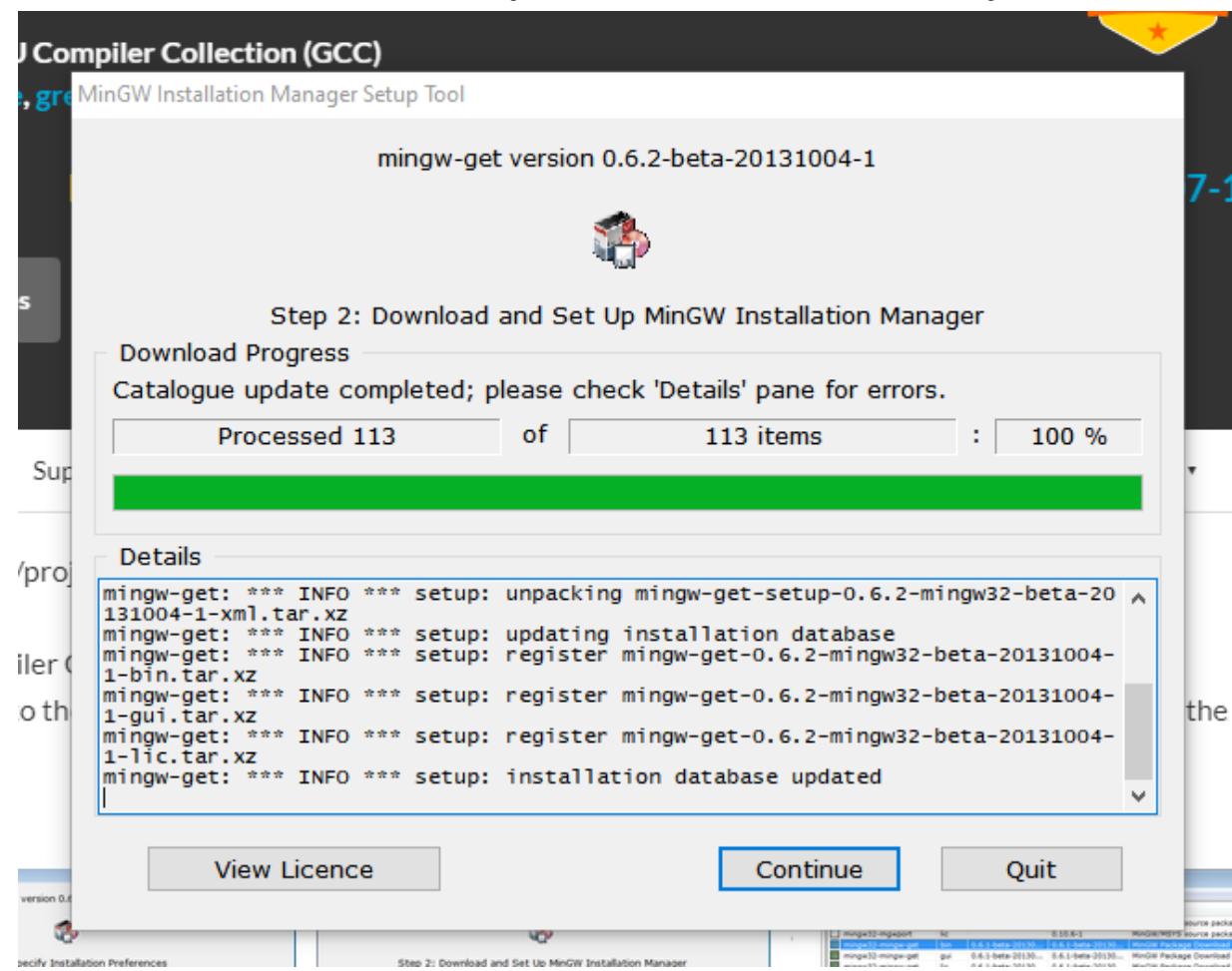
If you get a warning that MinGW is already installed, you can either reinstall and continue with this walkthrough or choose Run Now and skip to the steps below with the MinGW Install Manager.



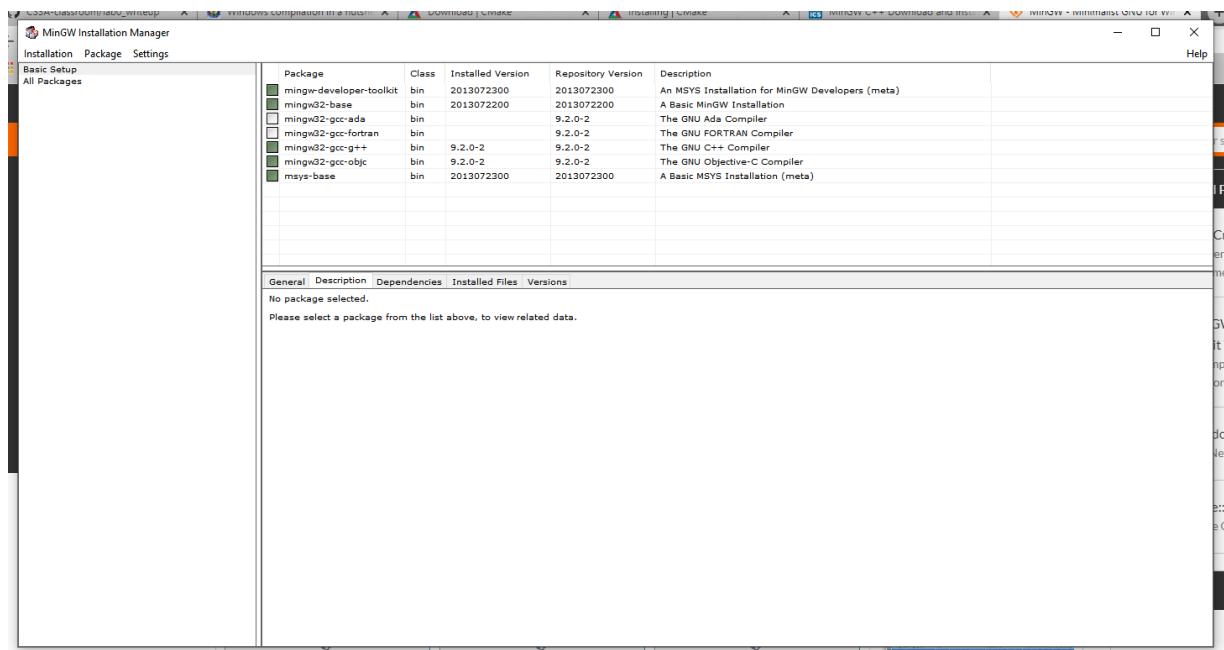
Installing..



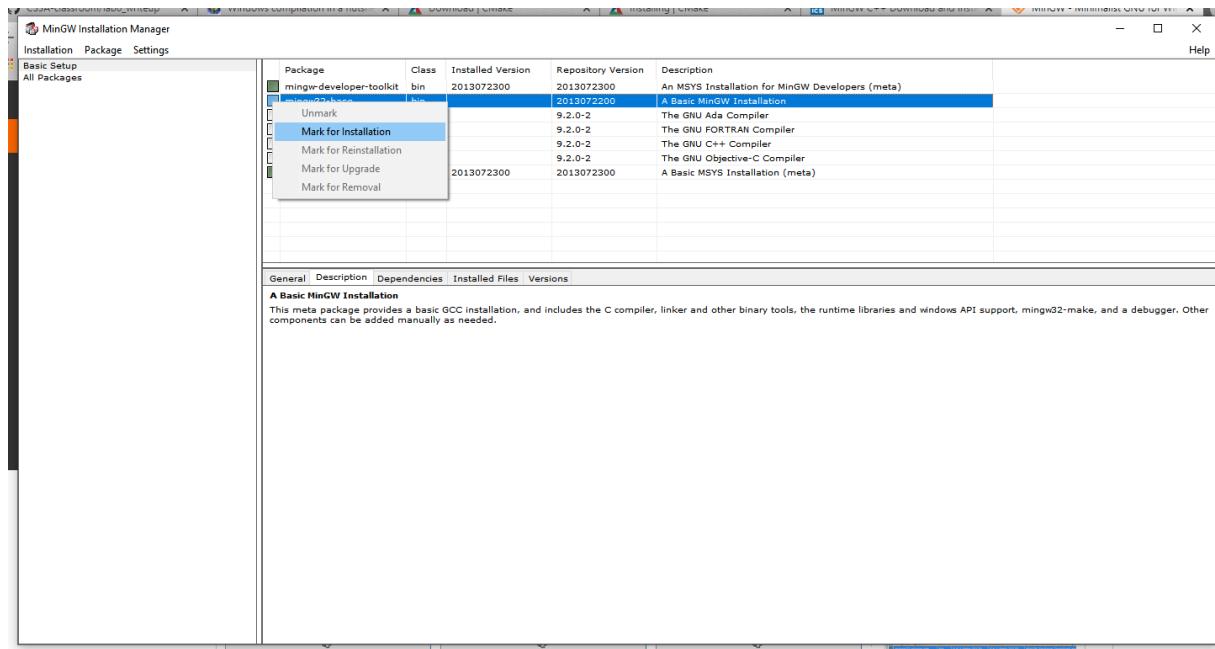
When it is done installing, click Continue.



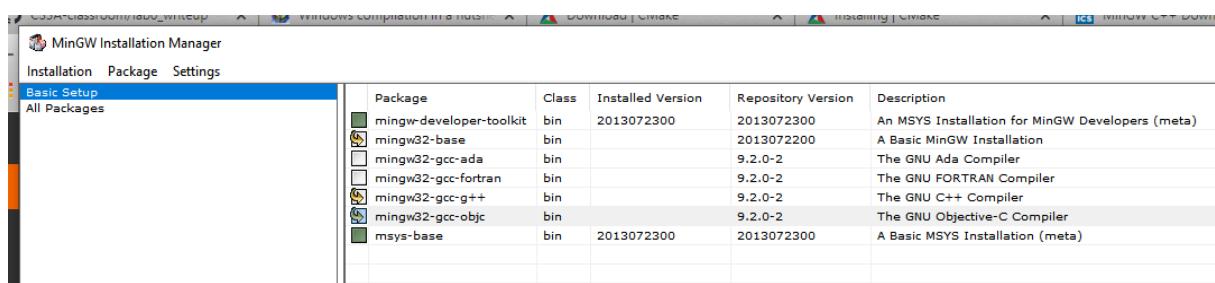
The MinGW Installation Manager should be opened up automatically. For this class, you will need mingw32-base, mingw-gcc-g++, and mingw23-gcc-objc.



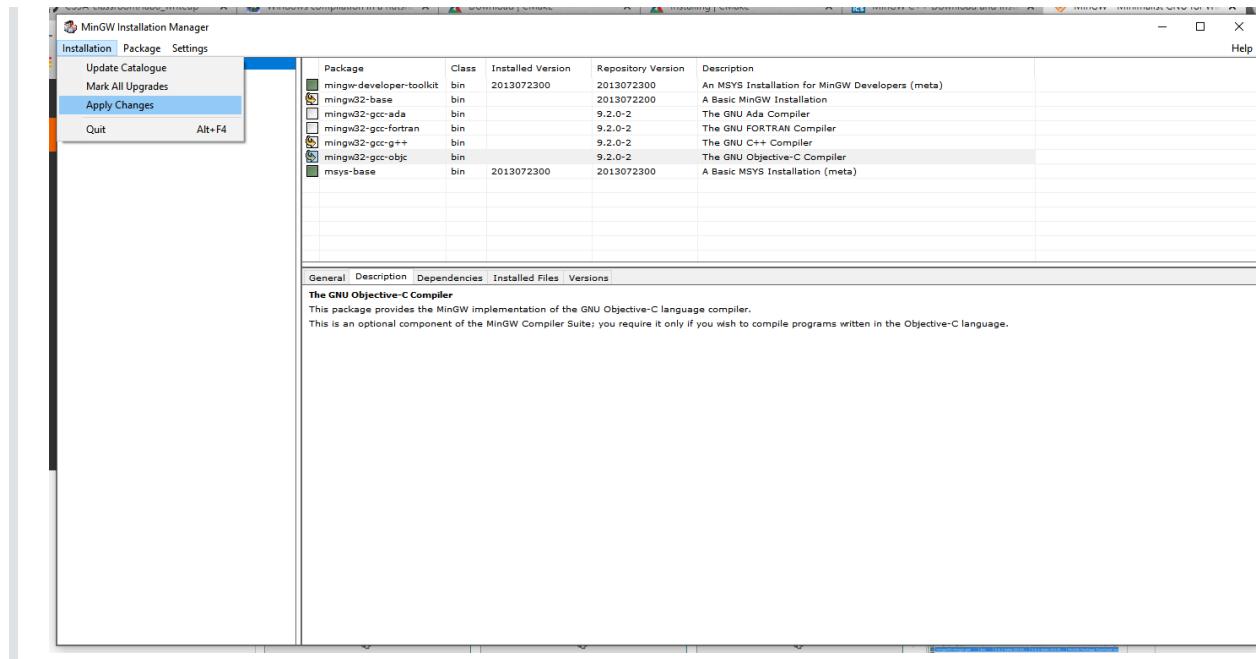
To select a package for installation, right click on it in the menu and select Mark for Installation.



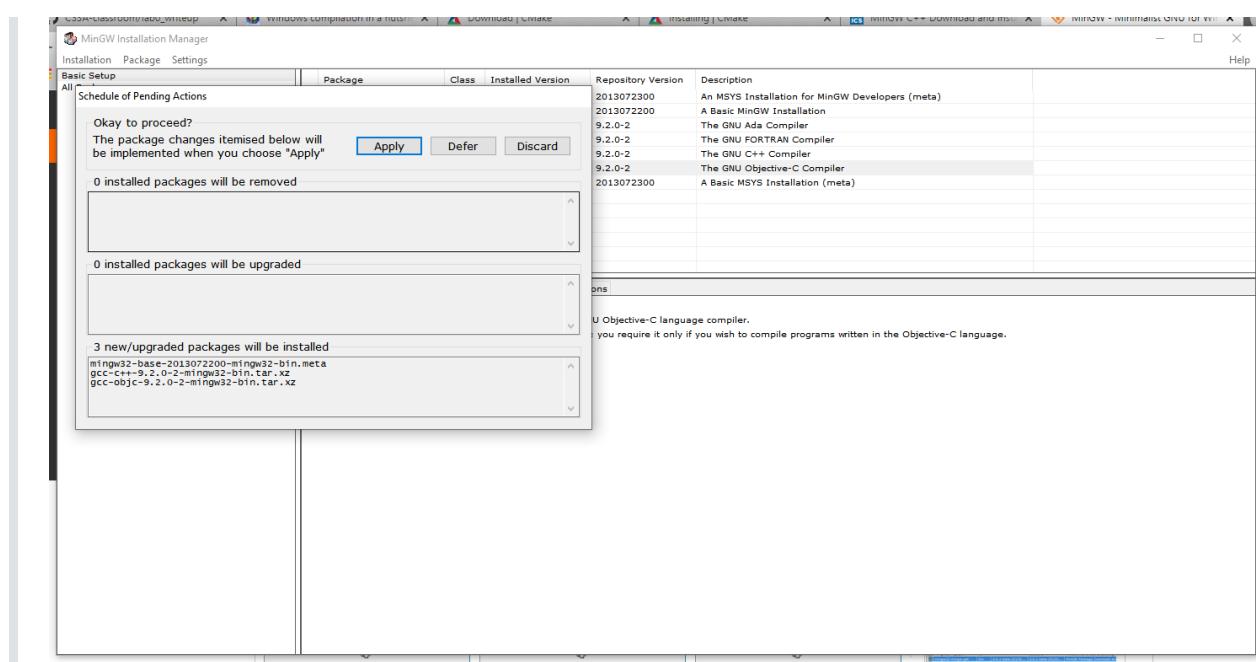
Marked packages will be selected like this:



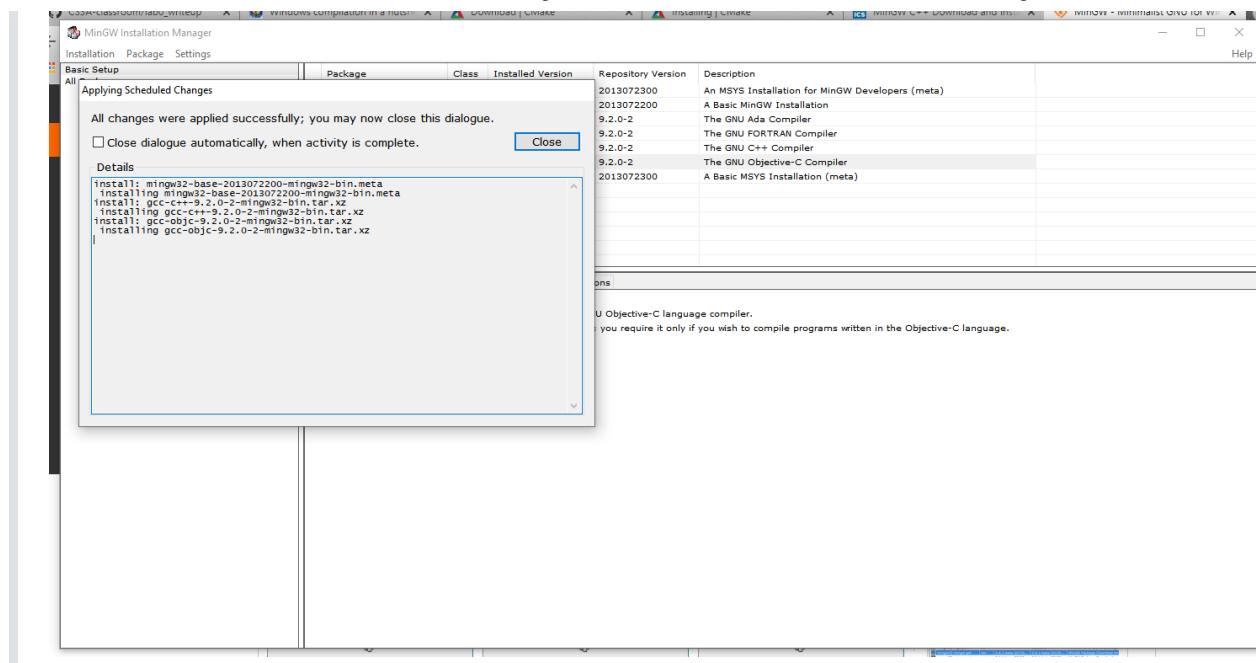
Once all necessary packages have been marked, go to Installations > Apply Changes to install.



Click Apply

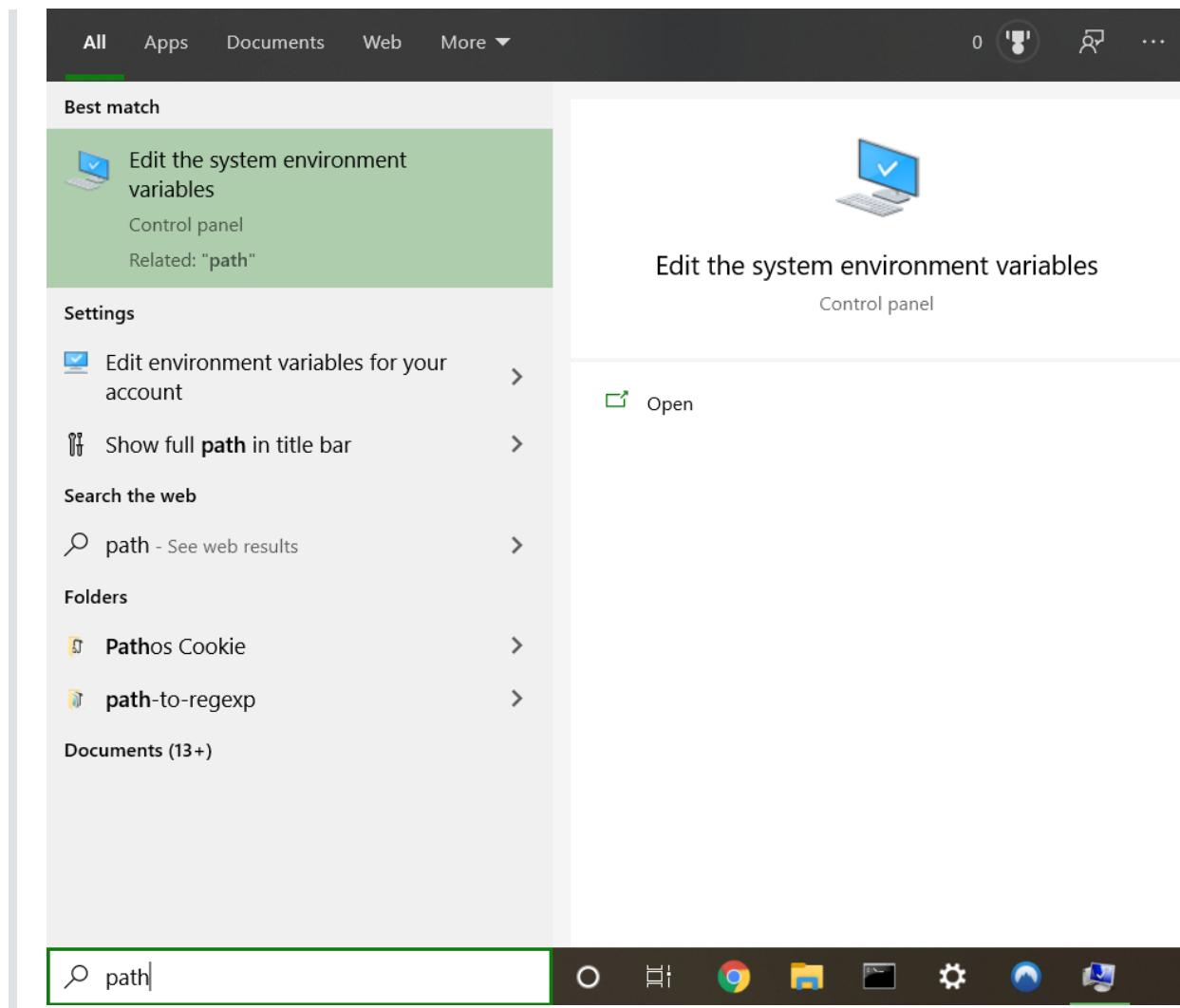


Once the changes are applied you may close the Installation Manager.

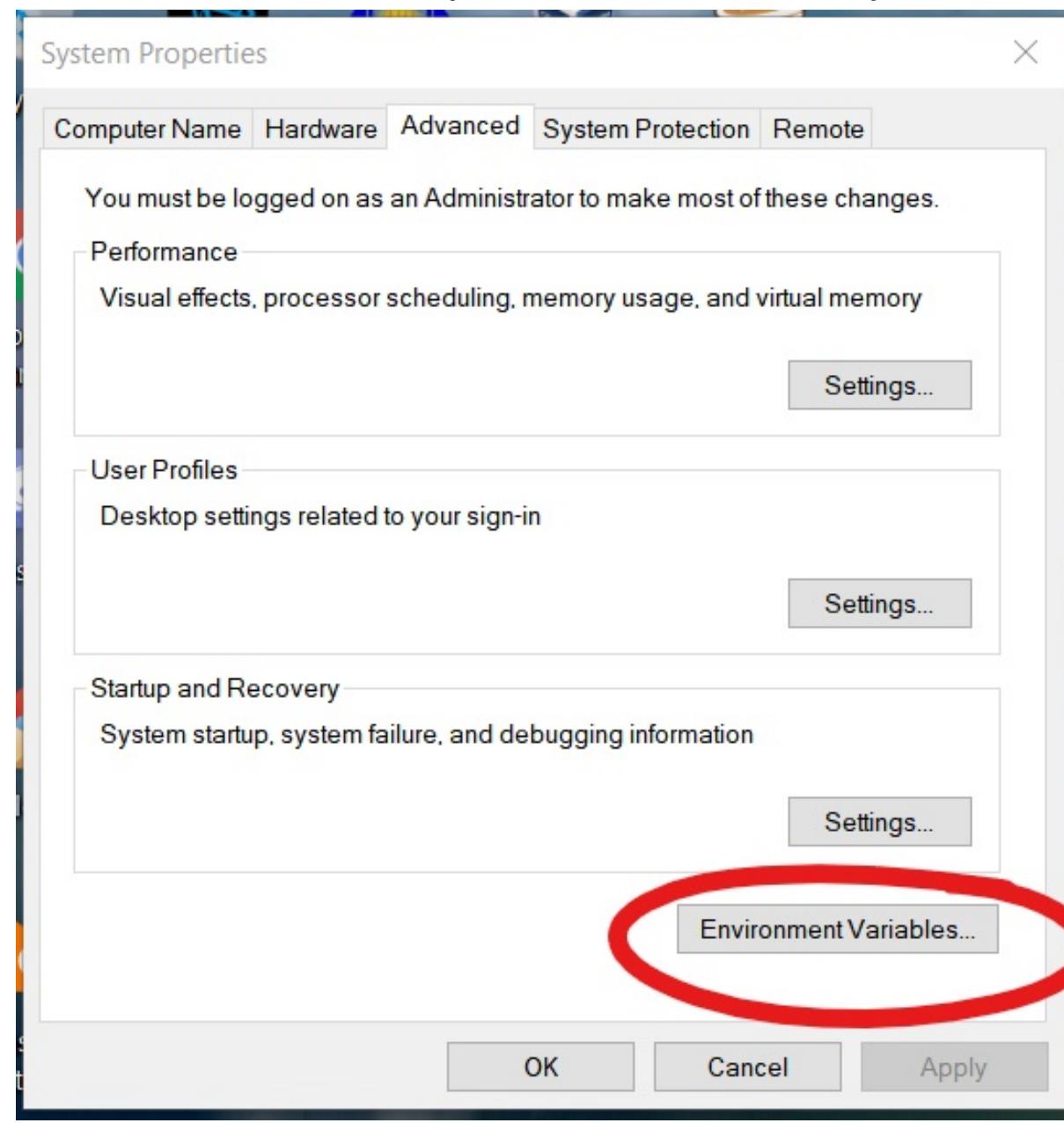


Add g++ as a System Variable and to your path

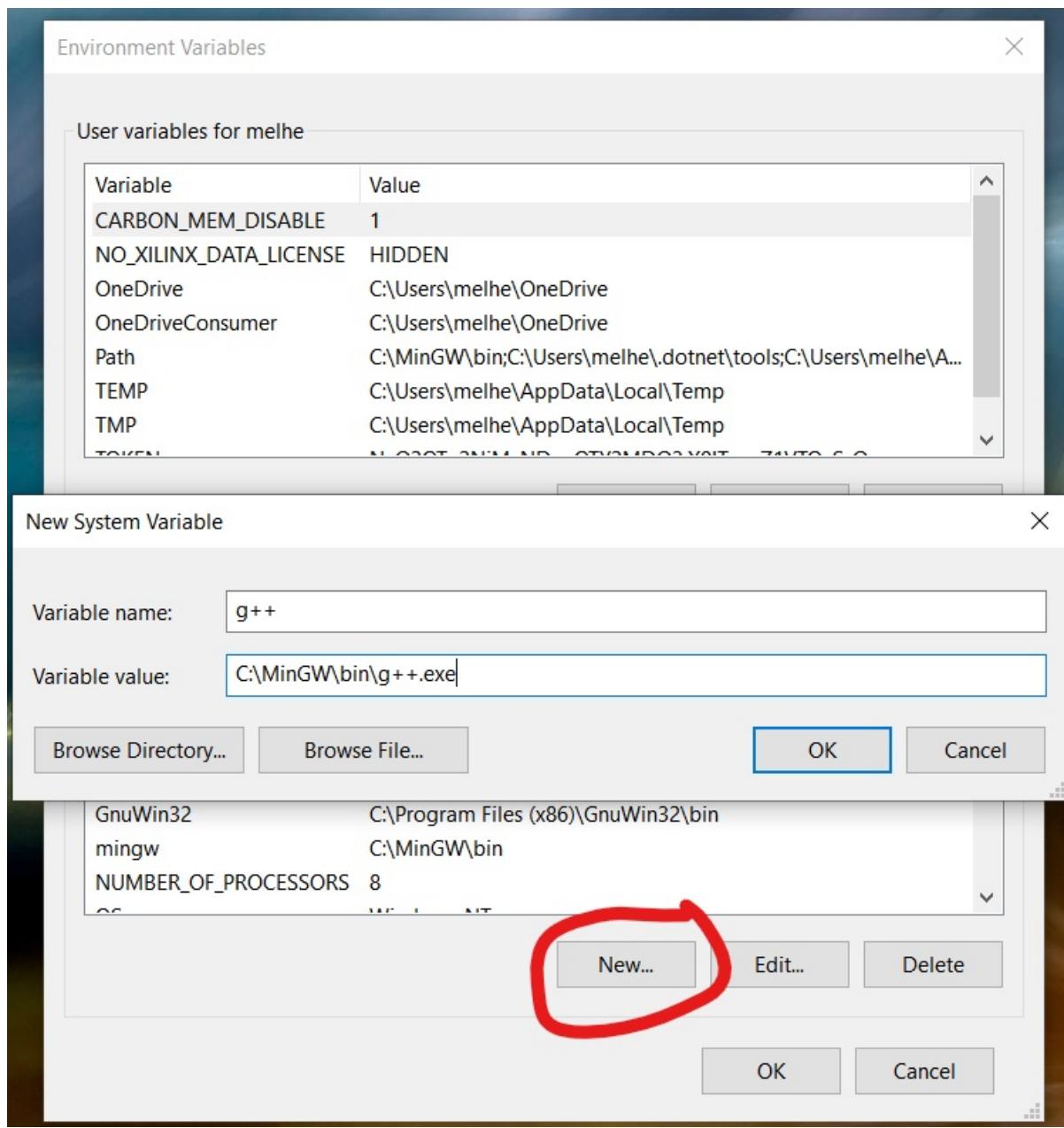
Search for "path" in the task bar search box. Open Edit the system environment variables.



Click on "Environment Variables..."



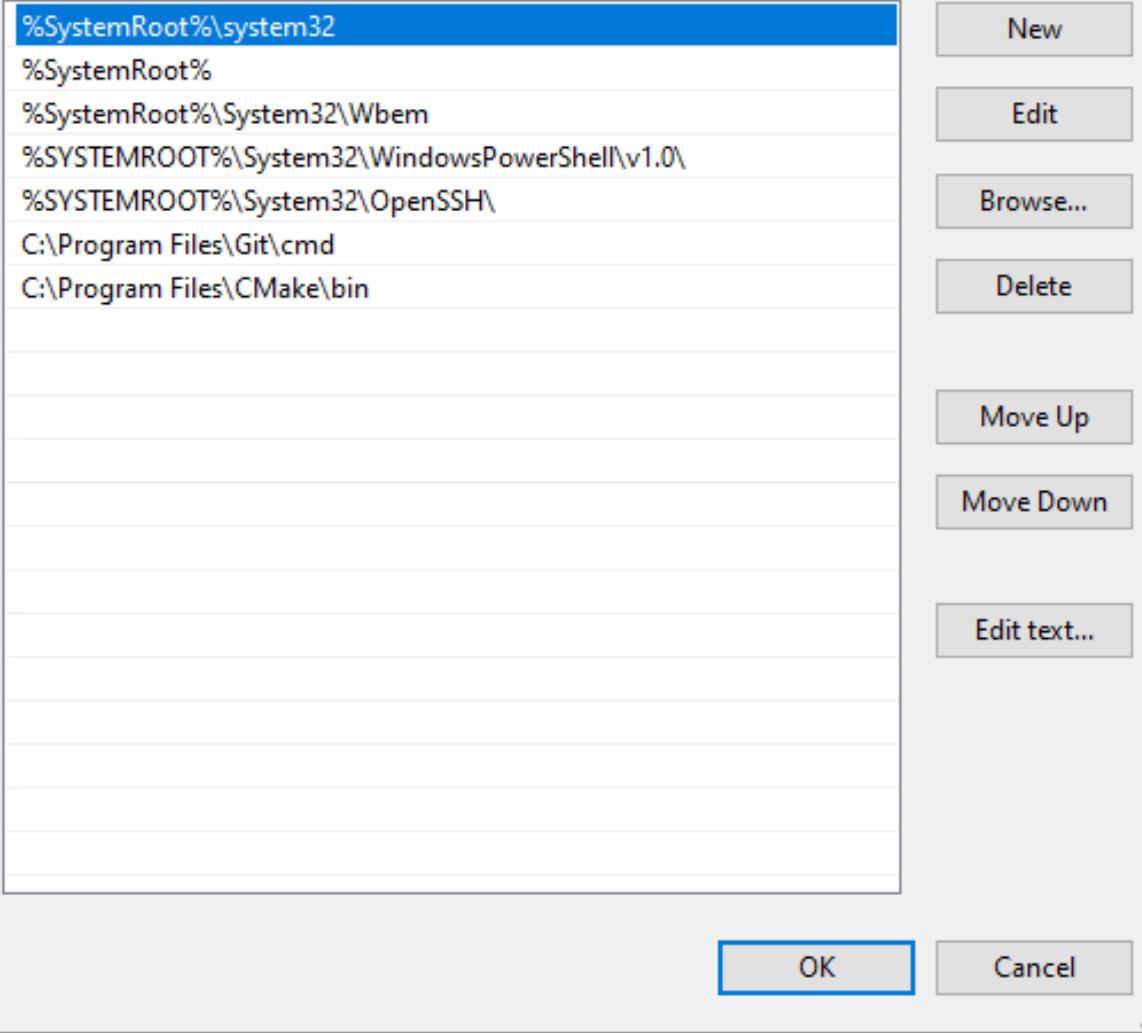
Add the path to the gcc and g++ executables to environment variables as shown below and press OK.



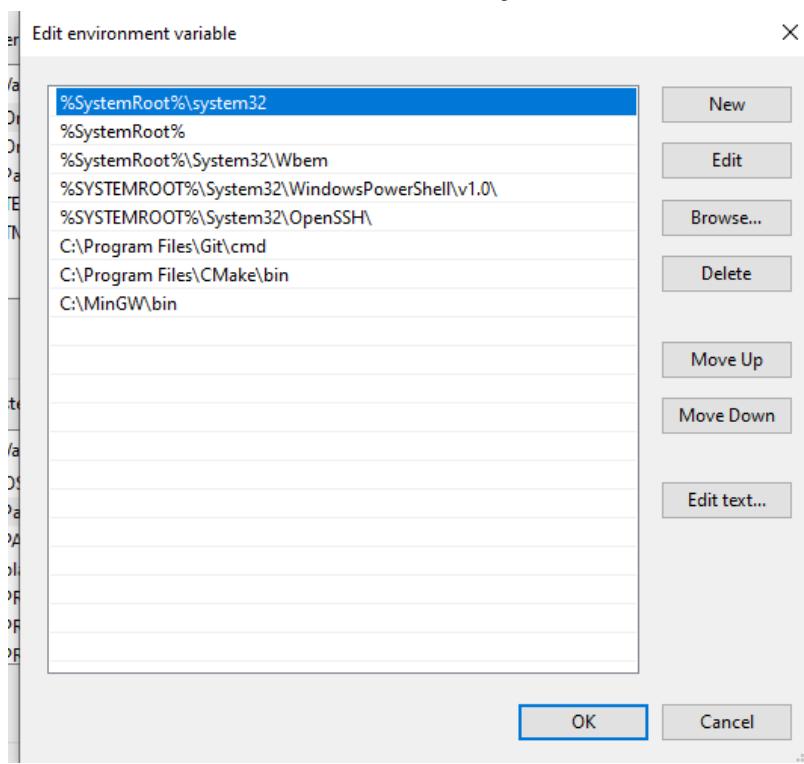
Next, double click on the **user variable** Path (in the top half of the window). A window like this should pop up:



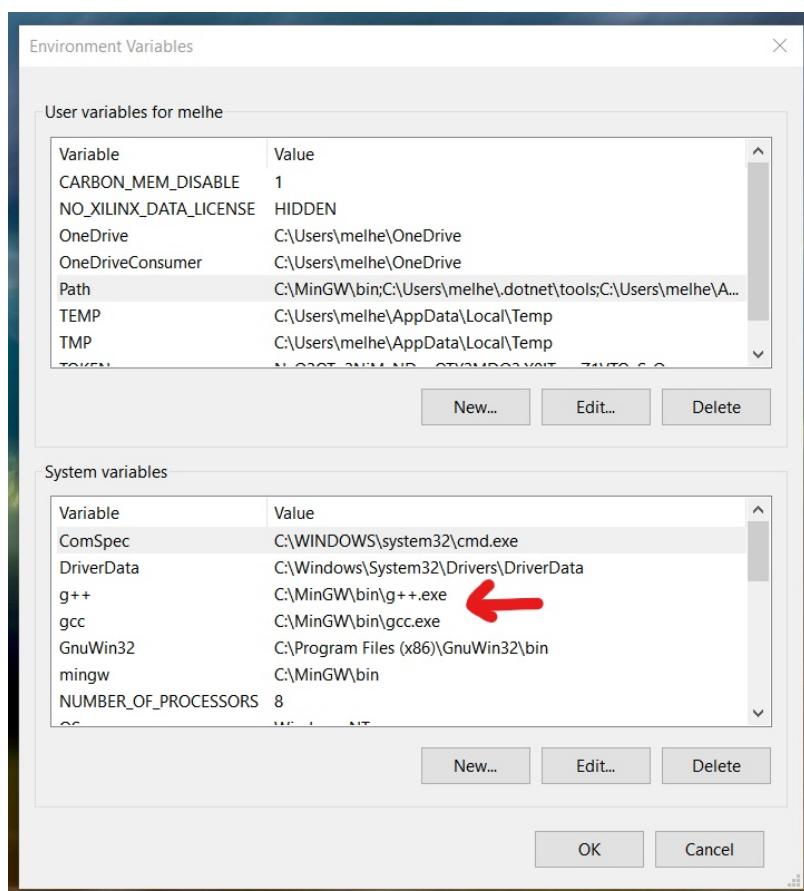
Edit environment variable



Click New, and enter C:\MinGW\bin to the text box. Click OK.

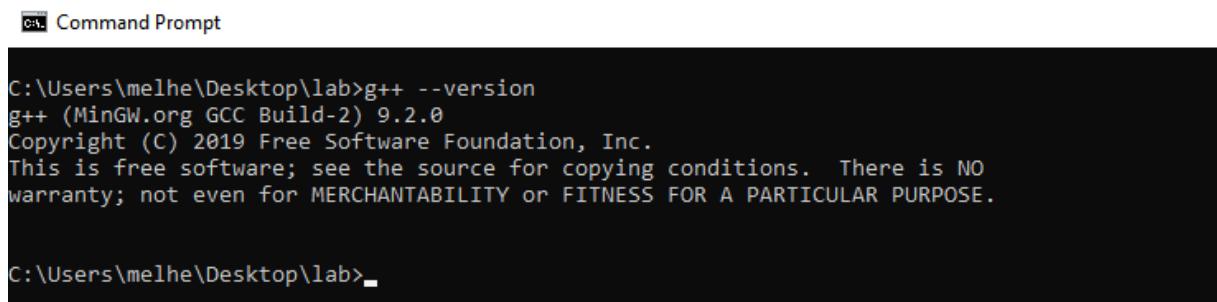


Your environment and user variables should look like this after you are done:



Check the version of the g++ again:

To make sure MinGW / g++ is intalled correctly, run `g++ --version` again. If the version output doesn't show, reboot your machine and try again.



```
cmd Command Prompt

C:\Users\melhe\Desktop\lab>g++ --version
g++ (MinGW.org GCC Build-2) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

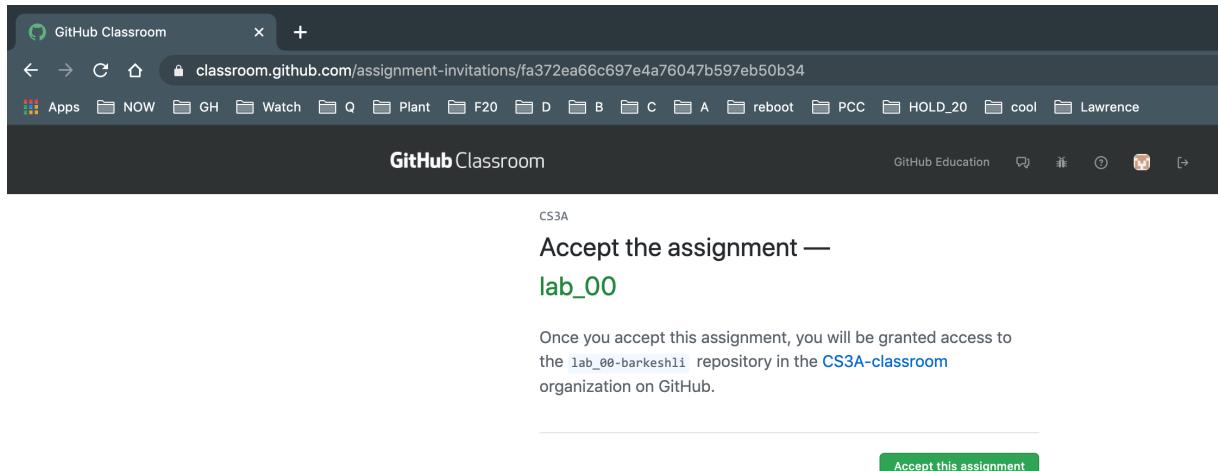
C:\Users\melhe\Desktop\lab>
```

■ Accept the assignment ■

Here is the assignment link for [CS3A](#) and here is the link for [CS8](#)

Accept assignment page:

Once you click on the [assignment link](#), we need you to *accept* the assignment. This will create a repo under your github username. But before you click and accept the assignment, let's look at a couple of things a bit more closely.



Your repo name:

This is your repo name. The name of the assignment followed by your github name.

ice you accept this is

the `lab_00-barkeshli` re

ganization on GitHub

Accept the assignment

You will accept the assignment by clicking the green button.

After you will be granted access to
your in the **CS3A-classroom**

Accept this assignment

Your assignment repo being created.

Once you accept the assignment, github will begin to create your assignment repo. You will see this page:

The screenshot shows a browser window for GitHub Classroom. The URL is classroom.github.com/assignment-invitations/fa372ea66c697e4a76047b597eb50b34/status. The page displays a message: "You accepted the assignment, lab_00. We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates." Below this, a note says: "Note: You may receive an email invitation to join CS3A-classroom on your behalf. No further action is necessary." To the right, there's a sidebar with a "Join the GitHub Student Developer Pack" section, which includes a "Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)". A blue "Apply" button is at the bottom of this sidebar.

Your assignment repo:

Give it a few seconds, and reload the page and you should see this:

The screenshot shows a browser window for GitHub Classroom. The URL is classroom.github.com/assignment-invitations/fa372ea66c697e4a76047b597eb50b34/status. The page displays a message: "You're ready to go!" with a hands icon. It states that the user accepted the assignment "lab_00". Below this, it says "Your assignment repository has been created:" followed by a link to the repository: https://github.com/CS3A-classroom/lab_00-barkeshli. A note at the bottom indicates that an email invitation may be sent to join the classroom.

look closer:

Take a closer look and you will see the link to your repo. Click it and you will find your assignment repo:

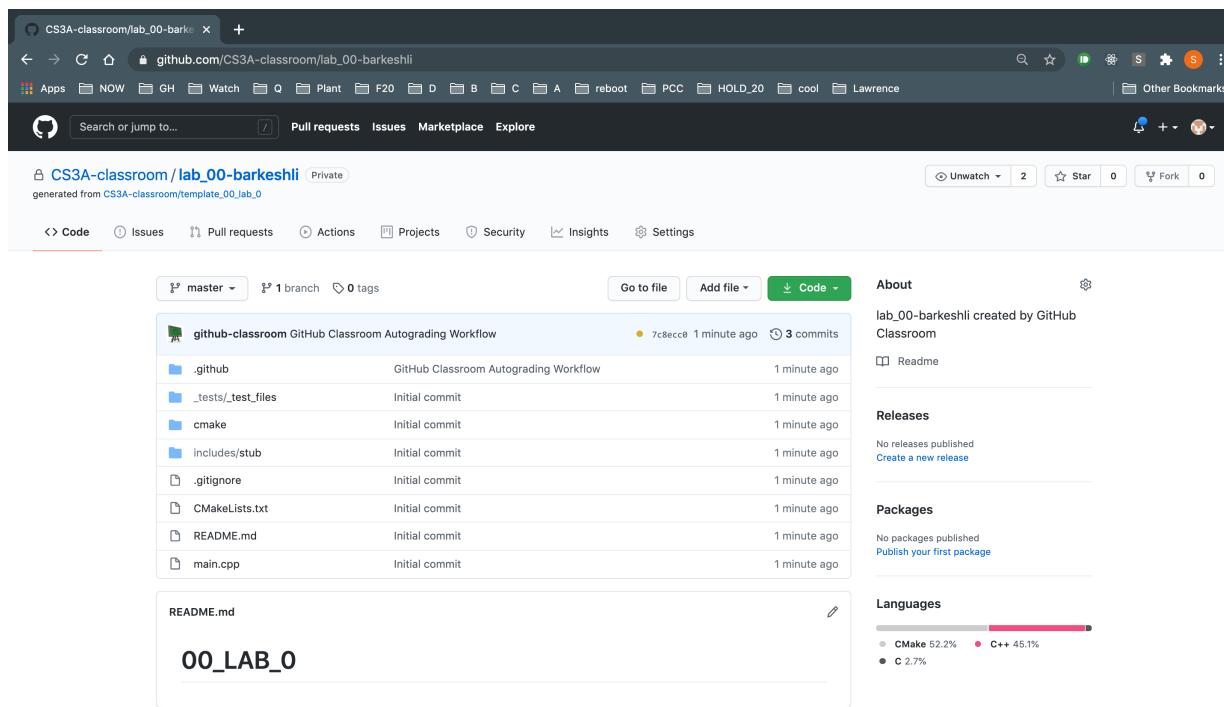
YOUR ASSIGNMENT REPOSITORY HAS BEEN CREATED:

 https://github.com/CS3A-classroom/lab_00-barkeshli

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

Your assignment repo:

Bookmark this page or know how to get here. We'll need to check in here soon.



The Code button:

The green button on the mid-right side that says **Code**, click it and that opens this box:

Click the little clipboard and that will copy the link into your clipboard so you can paste it in the next step. You will use this url to clone your repo:

The screenshot shows a GitHub repository page for 'CS3A-classroom/lab_00_barkeshli'. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. A dropdown menu is open under 'Code' with the following options:

- Clone**: Includes links for HTTPS, SSH, and GitHub CLI, with a URL field containing https://github.com/CS3A-classroom/lab_00_barkeshli and a copy icon.
- Open with GitHub Desktop**
- Download ZIP**

Below the dropdown, the text '1 minute ago' is visible. To the right of the dropdown, there are sections for 'About', 'Releases', and 'Packages'.

clone the assignment repo:

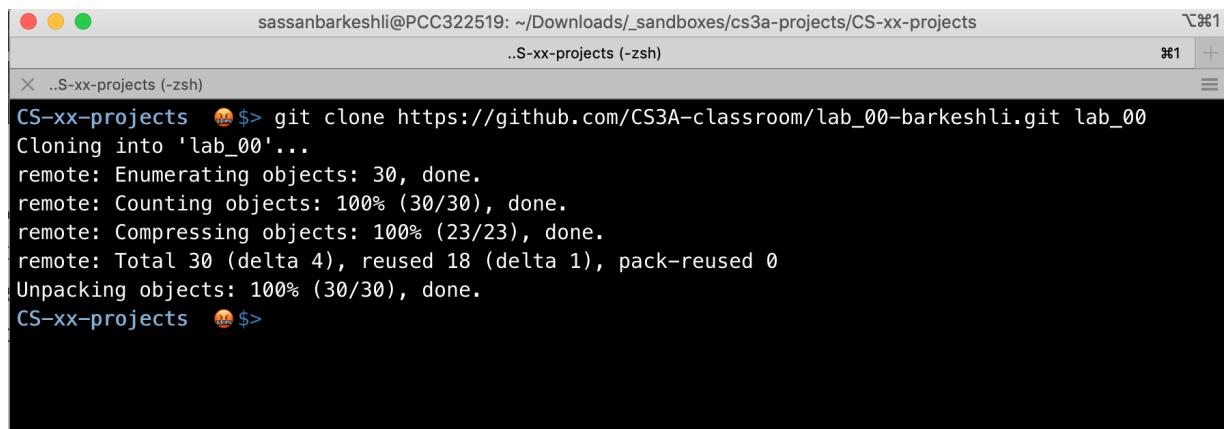
Before you can work on your project, you will need a local copy of the assignment. This is called **cloning** the repository.

First step is to `cd` into the folder where you will be storing all your projects.

`cd` into your projects folder and clone your project there.

and then: `git clone [clone link] [destination_folder]`

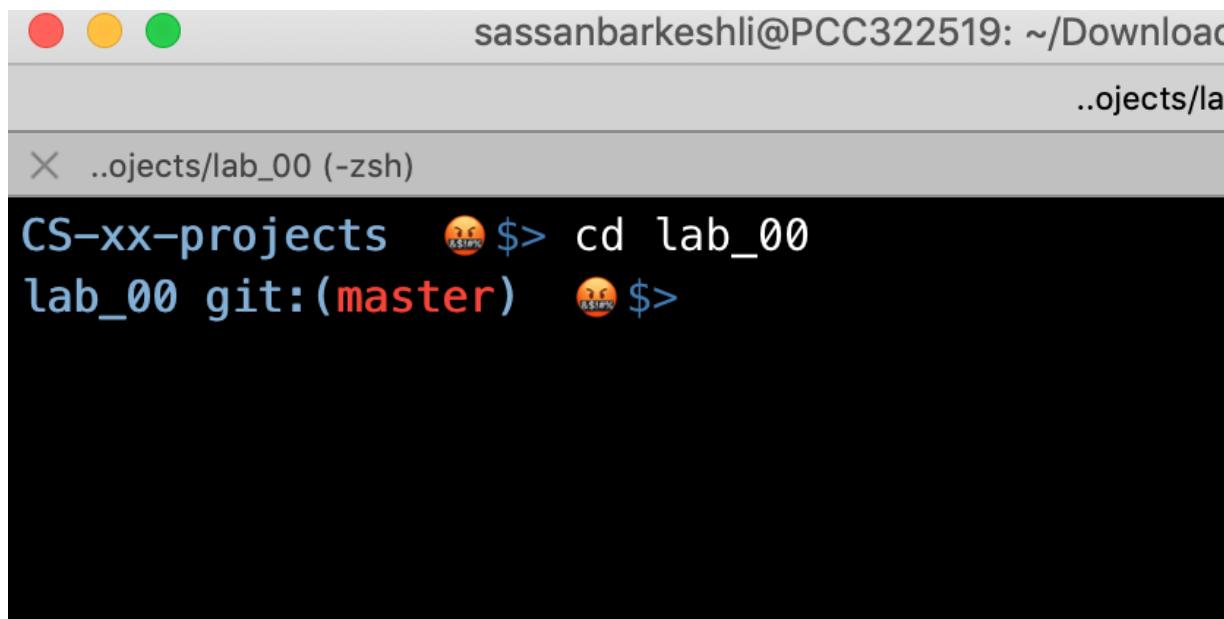
That's what's happening here. I don't like my folder name to be `lab_00_barkeshli`. I like `lab_00`, so, I give it the new name and that clones the project into a folder named `lab_00`



A screenshot of a terminal window titled "sassanbarkeshli@PCC322519: ~/Downloads/_sandboxes/cs3a-projects/CS-xx-projects ..S-xx-projects (-zsh)". The window shows the command "git clone https://github.com/CS3A-classroom/lab_00-barkeshli.git lab_00" being run. The output of the command is displayed, showing the progress of cloning the repository, including object enumeration, counting, compressing, and unpacking. The process is completed successfully.

```
git clone https://github.com/CS3A-classroom/lab_00-barkeshli.git lab_00
Cloning into 'lab_00'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 30 (delta 4), reused 18 (delta 1), pack-reused 0
Unpacking objects: 100% (30/30), done.
CS-xx-projects 😊$>
```

cd into this project folder:



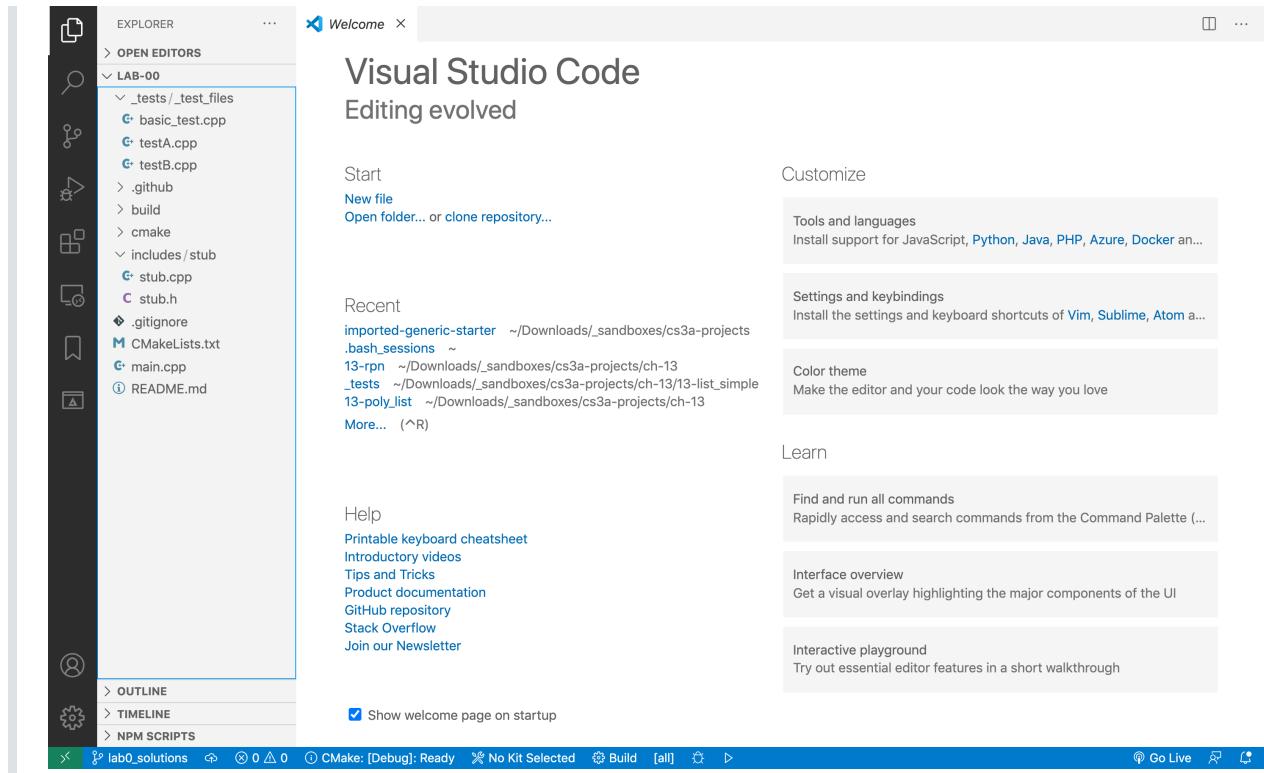
A screenshot of a terminal window titled "sassanbarkeshli@PCC322519: ~/Downloads/_sandboxes/cs3a-projects/CS-xx-projects ..objects/lab_00 (-zsh)". The window shows the command "cd lab_00" being run. The output shows the user has moved into the "lab_00" directory, indicated by the prompt "lab_00 git:(master) 😊\$>".

```
cd lab_00
lab_00 git:(master) 😊$>
```

We will be using VS Code to view and edit our code. Open your newly cloned project int VS Code.

Project Organization:

Once you have cloned the project and you open VSCode, this is what you will see:



File system:

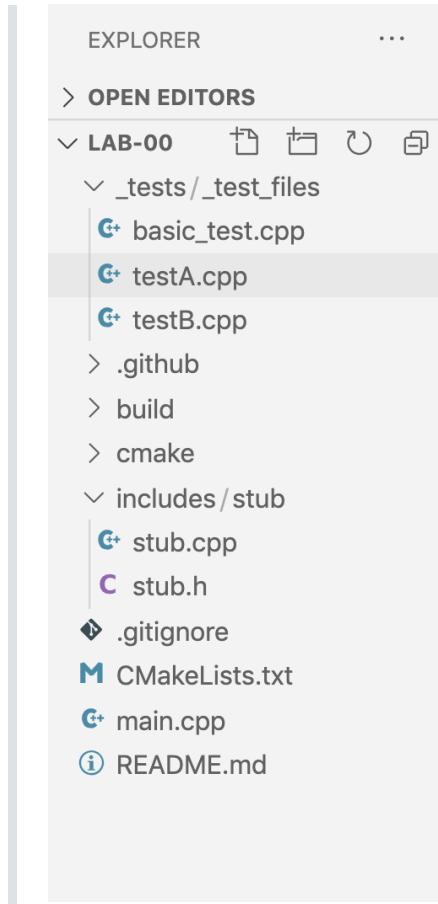
All the projects in this class will follow the same file organization.

On the left panel (Explorer,) you will find the `main.cpp` on the root folder, and the three most important folders in this directory:

`_tests` : which holds your google test files. The grader will run these files to obtain your score. You will ignore `testA.cpp` for the most part. The bulk of your work will be done in `testB.cpp`.

`includes` : contains a folder for each of the libraries / classes your project depends on. In this starter code, you only have a `stub` folder that contains `stub.h` and `stub.cpp`. These files are `#include`d in `testB.cpp`

`build` : is where you go to build and run your project. This is where all your compiled and executable files will end up.



A look at the test files:

basic_test.cpp: My sample test goes here.

This is the placeholder file for a sample test file you will be given for each and every project. The purpose of this file is to demonstrate the functionality of the project and for you to make sure that your function signatures and class declarations match the grader's expectations. (otherwise, your projects will not earn a score.)

testB.cpp: Your tests go here

This is the file that will contain your tests of your own functions and classes. All your test files that will demonstrate the correctness of your project are housed here. Part of your grade relies on the quality and success of the tests in this file.

```

EXPLORER ... _tests > _test_files > basic_test.cpp > ...
OPEN EDITORS ... _tests > _test_files > testB.cpp > ...
LAB-00 ... _tests > _test_files > testB.cpp > ...
    <_tests/_test_files
        basic_test.cpp
        testA.cpp
        testB.cpp
    > .github
    > build
    > cmake
    > includes/stub
        stub.cpp
        stub.h
    < .gitignore
    CMakeLists.txt
    main.cpp
    README.md

basic_test.cpp ... _tests > _test_files > basic_test.cpp > ...
1 #include "gtest/gtest.h"
2 #include <iostream>
3 #include <iomanip>
4 #include "../../includes/stub/stub.h"
5 using namespace std;
6 //-----
7 //      COPY BASIC_TEST INTO THIS FILE.
8 //          AND THEN,
9 //      DO NOT EDIT THIS FILE ANY FURTHER
10 //-----
11
12 bool basic_test(bool debug = true){
13     if (debug){
14         cout << "\nbasic test...\\n" << endl;
15     }
16     return true;
17 }
18
19 TEST(BASIC_TEST, BasicTest) {
20
21     //EXPECT_EQ(1, <your individual test functions
22
23     EXPECT_EQ(1, basic_test(false));
24 }
25
26
27
28
29 int main(int argc, char **argv) {
30     ::testing::InitGoogleTest(&argc, argv);
31     std::cout<<"\\n\\n-----running basic_test.c
32     return RUN_ALL_TESTS();

```

```

basic_test.cpp ... _tests > _test_files > testB.cpp > ...
1 #include "gtest/gtest.h"
2 #include <iostream>
3 #include <iomanip>
4 #include "../../includes/stub/stub.h"
5
6 bool test_stub(bool debug = false){
7     bool test = stub();
8     return test;
9 }
10
11 TEST(TEST_STUB, TestStub) {
12
13     //EXPECT_EQ(0, <your individual test functions
14
15     EXPECT_EQ(1, test_stub(false));
16 }
17
18
19
20
21 int main(int argc, char **argv) {
22     ::testing::InitGoogleTest(&argc, argv);
23     std::cout<<"\\n\\n-----running testB.cpp---
24     return RUN_ALL_TESTS();
25 }
26
27
28
29 int main(int argc, char **argv) {
30     ::testing::InitGoogleTest(&argc, argv);
31     std::cout<<"\\n\\n-----running basic_test.c
32     return RUN_ALL_TESTS();

```

CMakeLists.txt

The github grader as well as your local Mac or linux systems will use the CMakeLists.txt file to build your project.

List your cpp files here

the `CMakeLists.txt` file is what the `cmake` program looks at to know how to build your project. How the pieces fit together.

When you submit your code, you need to tell it what files are needed to build the test executables.

The grader will also use this file to build your project on the server side (once you submit - push your projects to github)

Please note that you will **only** make changes to the bottom half of this file. It's worth mentioning that every `.cpp` file that is used in any of your test files (main, basic_test, testA, testB) will have to be listed here. Notice how the `stub.cpp` is listed under `ADD_EXECUTABLE(testB...)`

```

EXPLORER      ...
OPEN EDITORS  M CMakeLists.txt X
              M CMakeLists.txt
              23
              24 # -----
              25 #                               DO NOT EDIT
              26 #                               ANYTHING ABOVE THIS LINE
              27 #
              28
              29
              30 # ADD_SUBDIRECTORY(googletest)
              31
              32 ADD_EXECUTABLE(main
              33     main.cpp
              34     includes/stub/stub.cpp
              35 )
              36
              37 ADD_EXECUTABLE(basic_test
              38     _tests/_test_files/basic_test.cpp
              39 )
              40 ADD_EXECUTABLE(testA
              41     _tests/_test_files/testA.cpp
              42 )
              43
              44 ADD_EXECUTABLE(testB
              45     _tests/_test_files/testB.cpp
              46     includes/stub/stub.cpp
              47 )
              48
              49 TARGET_LINK_LIBRARIES(basic_test gtest)
              50 TARGET_LINK_LIBRARIES(testA gtest)
              51 TARGET_LINK_LIBRARIES(testB gtest)
              52
              53
  
```

stub.h, stub.cpp

Not too much to see here. The stub is used in `testB` to demonstrate how a function will be tested by the googletest framework in `testB.cpp`. All your functions and classes will be housed under their own folder (`stub/`, `array_functions/`, `vector/`, etc.) which will, in turn, go under the `includes/` folder.

```

C stub.h X
includes > stub > C stub.h > ...
1 #ifndef B_STUB_H
2 #define B_STUB_H
3 bool stub(); //stub to show how included li
4 |   |   |   |   |   |   // into our projects
5
6 #endif

C stub.cpp X
includes > stub > C stub.cpp > ...
1 #include <iostream>
2 #include <iomanip>
3 #include "stub.h"
4
5 bool stub(){
6     return true;
7 }

```

Open terminal:

If you are using VSCode, you can open the terminal by pressing [ctrl][`]

[`] is the key in the top left of the keyboard under [~]

Using the terminal in this way is very convenient.

```

array_functions.cpp - lab-00 - Visual Studio Code
File Edit Selection View Go Run Terminal Help array_functions.h X ...
EXPLORER OPEN EDITORS
LAB-00 _tests/_test_files basic_test.cpp array_functions.h ...
includes > array_functions > C array_functions.h > ...
1 #ifndef ARRAY_FUNCTIONS_H
2 #define ARRAY_FUNCTIONS_H
3
4 #include <iostream>
5 using namespace std;
6
7 void _array_init(int a[], int size, int x=0);
8
9 void _append(int a[], int& size, int append_me);
10
11 int _find(const int a[], int size, int find_me);
12
13 int& _at(int a[], int size, int pos);
14
15 ostream& _print_array(const int a[], int size, ostream& outs = cout
16
17
18 #endif // ARRAY_FUNCTIONS_H

```

```

array_functions.h X
includes > array_functions > C array_functions.cpp > ...
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x=0)
5 {
6
7 }
8
9 void _append(int a[], int& size, int append_me)
10 {
11
12 }
13
14 int _find(const int a[], int size, int find_me)
15 {
16
17     return 0;
18 }
19
20 int& _at(int a[], int size, int pos)
21 {
22     return a[0];
}

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\mehm\Desktop\lab-00>

■ Quick edit, status , add , commit & push ■

One of the main tasks in this class is tracking changes made to the project. We need to know what happened when and what changed. This is both for your peace of mind (helps you not lose your project accidentally,) and for me to track your progress throughout the course.

We use **git** to track changes.

Here, we will make some small change to one of our files and walk through the entire process of tracking that change with `git` , just so you can get used to it.

git status

let's run `git status` . look at the response: It says there are not changes as of yet. That's correct, isn't it. We have not made any changes.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files and folders, including 'LAB-00' which contains '_tests__test_files', 'basic_test.cpp', 'testA.cpp', 'testB.cpp', '.github', 'build', 'cmake', 'includes', '.gitignore', 'CMakeLists.txt', 'main.cpp', and 'README.md'. The main editor area shows the 'README.md' file with the following content:

```

# 00_LAB_0
Sassan Barkeshli

```

The terminal below shows the output of a 'git status' command:

```

PS C:\Users\melhe\Desktop\lab-00> git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
PS C:\Users\melhe\Desktop\lab-00>

```

Edit the README.md

Enter your name in the README.md and save it.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files and folders, including 'LAB-00' which contains '_tests__test_files', 'basic_test.cpp', 'testA.cpp', 'testB.cpp', '.github', 'build', 'cmake', 'includes', '.gitignore', 'CMakeLists.txt', 'main.cpp', and 'README.md'. The main editor area shows the 'README.md' file with the following content:

```

# 00_LAB_0
Sassan Barkeshli

```

The terminal below shows the output of a 'git status' command:

```

PS C:\Users\melhe\Desktop\lab-00> git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
PS C:\Users\melhe\Desktop\lab-00>

```

git status again:

if you run `git status` again, you will see that git has kept track of our changes. this time, it says that the file `README.md` has changed. But it says that the file is not *staged* for commit.

Think of it this way... There are four zones:

[changes not staged]

---- `git add` ---->

[staged changes, but not committed]

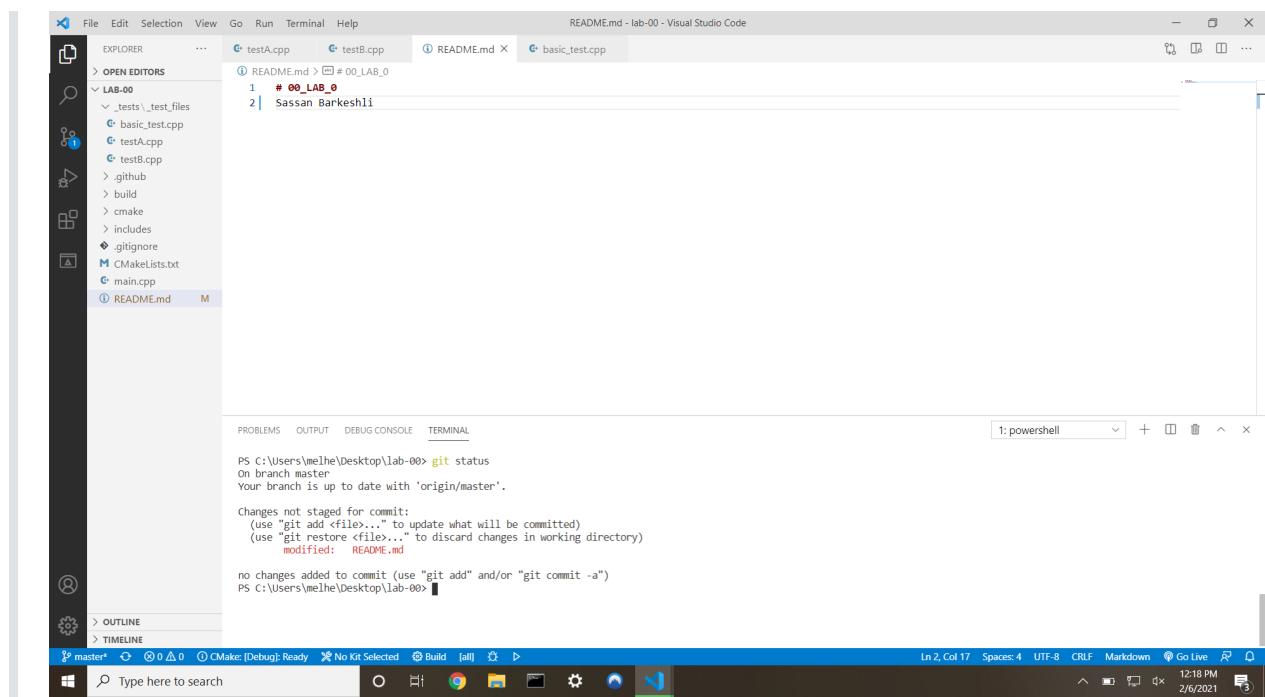
---- `git commit` ---->

[committed changes]

---- `git push` ---->

[pushed, or synched with remote site]

You go from zone to zone by the commands listed above.



git add and then, git status again:

In order to *stage* the changes for `commit`, Enter `git add README.md`. This will stage the file.

do `get status` again. Now, you see that `README.md` has been *staged*

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure for "LAB-00" containing files like testA.cpp, testB.cpp, README.md, and CMakeLists.txt.
- Terminal:** Displays the command `git status` and its output:


```
PS C:\Users\melhe\Desktop\lab-00> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified: README.md

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\melhe\Desktop\lab-00> git add README.md
PS C:\Users\melhe\Desktop\lab-00> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified: README.md

PS C:\Users\melhe\Desktop\lab-00>
```
- Bottom Status Bar:** Shows the current file is "README.md", the line and column numbers (Ln 2, Col 17), and the date and time (2/6/2021, 12:19 PM).

commit your changes:

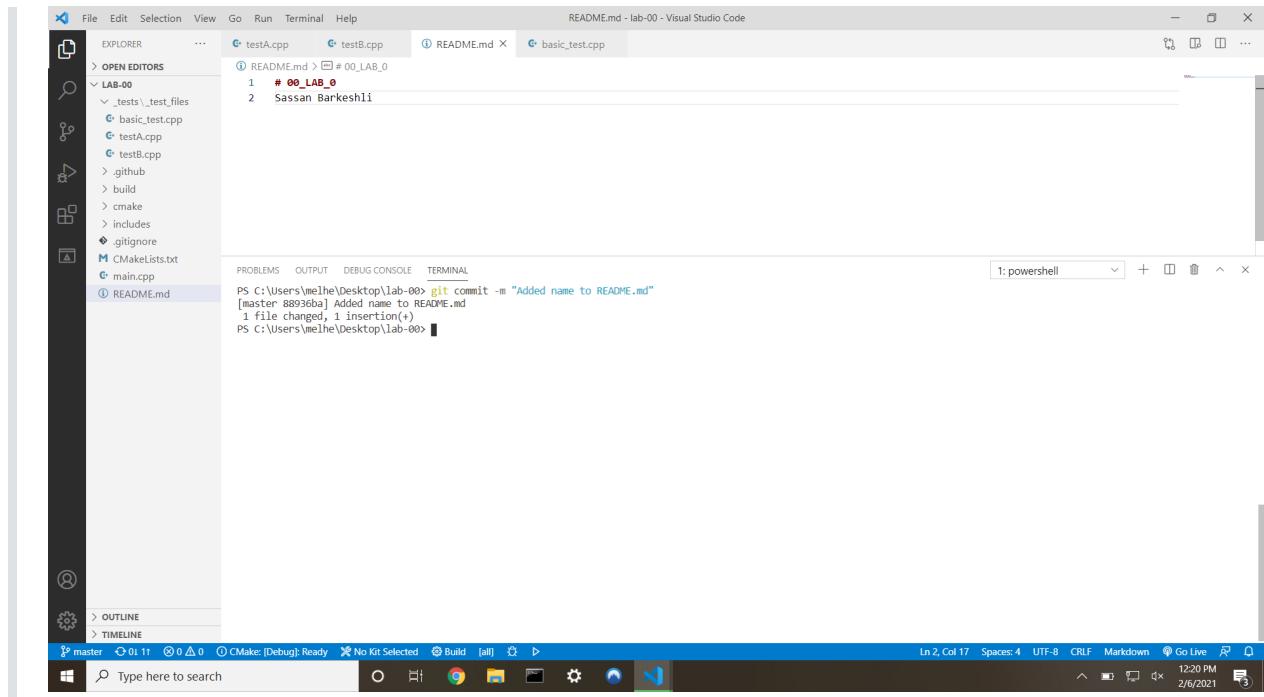
You `commit` your changes to permanently record these changes to `git`. If somehow you ruin your project (as we all have done from time to time,) you can *revert* to this state of your project.

`commit` ting is like *saving* your changes to `git`

Enter `git commit -m "[explain what you just did, in your own words]"`

Here, `-m` is a switch that tells `git` that you will be typing a message to document what this `commit` was for. We want this message to be concise and descriptive of the work that is being recorded.

Try typing in your own message in your own words.



A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a project structure with files like testA.cpp, testB.cpp, README.md, and CMakeLists.txt. The terminal tab is active, displaying a git commit command:

```
PS C:\Users\melhe\Desktop\Lab-00> git commit -m "Added name to README.md"
[master 88936ba] Added name to README.md
 1 file changed, 1 insertion(+)
PS C:\Users\melhe\Desktop\Lab-00>
```

Has Github assignment repo changed?

Take a look at your assignment repo on your browser. You will see that even though you have committed your changes, Github does not know about them!

`commit`ting only records your changes on your local machine.

What is my local branch name?

If we want Github to know about our changes, we must `push` them to Github. But before you do, let's find out what our current *branch* is.

For me, the branch name is `master` as evidenced by these images.

Your bash prompt on your terminal:

A screenshot of the Visual Studio Code interface. The left sidebar shows a project structure for 'LAB-00' with files like testA.cpp, testB.cpp, README.md, and CMakeLists.txt. The main editor area has two tabs open: 'README.md' and 'basic_test.cpp'. The terminal tab at the bottom is active, showing the command 'git branch' and its output: '* master'. The status bar at the bottom right shows the date and time as 2/6/2021 12:23 PM.

Zoomed in:

`git branch` will tell you the name of our local branch. Try it.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\melhe\Desktop\lab-00> git branch
* master
```

git push :

Since my branch name is `master`, I will issue the command `git push origin master` which means push to `origin`, which is my github remote name, from `master` which is my local branch name. For you, your branch name might be `main`, so, your command will be `git push origin main`:

If your command is successful, you will get a response similar to this:

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of a GitHub repository named 'LAB-00'. The repository contains files like 'testA.cpp', 'testB.cpp', 'README.md', and 'CMakeLists.txt'. The 'README.md' file is currently open in the editor, displaying the following content:

```
# 00_LAB_0
Sassan Barkeshli
```

Below the editor, the terminal window shows the output of a git push command:

```
PS C:\Users\melhe\Desktop\lab-00> git branch
* master
PS C:\Users\melhe\Desktop\lab-00> git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 29 bytes | 145.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1) completed with 1 local object.
To https://github.com/CS3A-classroom/lab_00-melh23.git
  e462bb4..88996ba master -> master
PS C:\Users\melhe\Desktop\lab-00>
```

Now, Github knows about our changes:

Take a look at your project repo on Github. See how the changes you made to README.md shows on your repo:

CS3A-classroom / lab_00-barkeshli (Private)

generated from CS3A-classroom/template_00_lab_0

Code Issues Pull requests Actions Projects Security Insights

master Go to file Add file Code

barkeshli added name to README.md 1 hour ago 4

📁 .github	GitHub Classroom Autograding Workflow	5 hours ago
📁 _tests/_test_files	Initial commit	5 hours ago
📁 cmake	Initial commit	5 hours ago
📁 includes/stub	Initial commit	5 hours ago
📄 .gitignore	Initial commit	5 hours ago
📄 CMakeLists.txt	Initial commit	5 hours ago
📄 README.md	added name to README.md	1 hour ago
📄 main.cpp	Initial commit	5 hours ago

README.md

00_LAB_0

Sassan Barkeshli

About lab_00-barkeshli created by GitHub Classroom

Readme

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages

CMake 52.2%
C++ 45.1%
C 2.7%

© 2021 GitHub, Inc. Terms Privacy Security Status Docs

commit and push often:

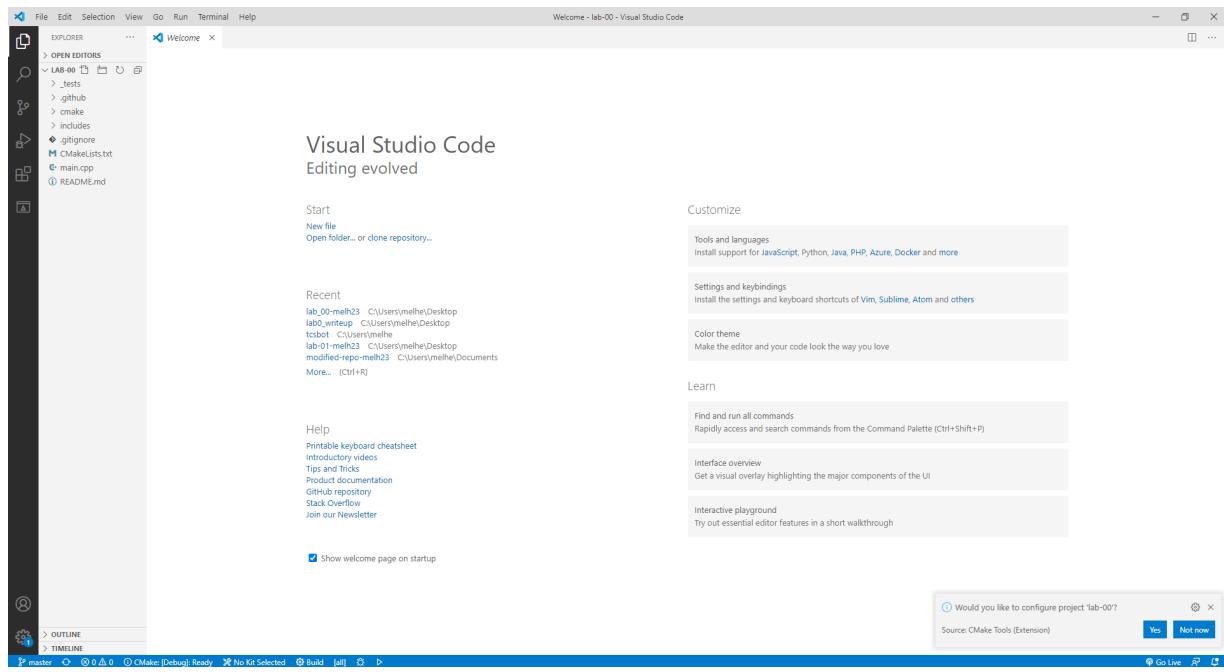
Make sure you `commit` your changes after completion of **every** task. Remember, `commits` are the record of your progress. You will be graded on your commits. And push your changes frequently.

Bonus: `git log`

Enter `git log` and see what the response is. What does this command do?

■ build and run walkthrough ■

Once your project is open in VS Code, we will do a quick exercise to show you how to run it.



Build the googletest framework

Before you can run your tests on your program, you must build the googletest framework. This has to be done only once per project **before** you do anything else.

To simplify this process, and the building of your project, a pair of batch files have been written.

The two .bat files you will use to build and compile your project: gtest.bat and build.bat will be included with your startup code for every project.

Navigate into the build folder and run gtest.bat by entering `. \gtest.bat` into the terminal. This will run a series of commands that build the googletest framework for you. This process creates a bunch of files and you do not want these files in the root folder of your project. Running the `gtest.bat` file from `build/` will make sure all your auxiliary files are created inside the `build/` folder. The output should look like this.

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Terminal Tab:** gtest.bat - lab-00 - Visual Studio Code
- Terminal Content:**

```

git clone https://github.com/google/googletest.git
cd googletest && mkdir build && cd build && cmake -G "MinGW Makefiles" .. && make && cd ../../

PS C:\Users\melhe\Desktop\lab-00> cd build
PS C:\Users\melhe\Desktop\lab-00\build> .\gtest.bat

C:\Users\melhe\Desktop\lab-00\build>cd googletest && mkdir build && cd build && cmake -G "MinGW Makefiles" .. && make && cd ../../

-- The C compiler identification is GNU 9.2.0
-- The CXX compiler identification is GNU 9.2.0
-- Detecting C compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/MinGW/bin/g++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/MinGW/bin/g++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: C:/Program Files (x86)/IronPython 2.7/ipy64.exe (found version "2.7.5") found components: Interpreter
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/melhe/Desktop/lab-00/build/googletest/build
Scanning dependencies of target gtest
[ 12%] Building CXX object googletest/CMakeFiles/gtest.dir/src/gtest-all.cc.obj
[ 25%] Linking CXX static library ..\lib\libgtest.a
[ 25%] Built target gtest
Scanning dependencies of target gmock
[ 37%] Building CXX object googletest/CMakeFiles/gmock.dir/src/gmock-all.cc.obj
[ 50%] Linking CXX static library ..\lib\libgmock.a
[ 50%] Built target gmock
Scanning dependencies of target gmock_main
[ 62%] Building CXX object googletest/CMakeFiles/gmock_main.dir/src/gmock_main.cc.obj
[ 75%] Linking CXX static library ..\lib\libgmock_main.a
[ 75%] Built target gmock_main
Scanning dependencies of target gtest_main
[ 87%] Building CXX object googletest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.obj
[100%] Linking CXX static library ..\lib\libgtest_main.a
[100%] Built target gtest_main
PS C:\Users\melhe\Desktop\lab-00\build>

```
- Bottom Status Bar:** master* CMake: Ready No Kit Selected Build [all]

build.bat needs to be run every time you make changes to your code. It creates executable files that you can run in the terminal. Run build.bat by entering `.\build.bat` into the VS Code terminal. It will create 3 executable files: testA.exe, testB.exe, and basic_test.exe.

Once your executables have been sucessfully created, you can run the basic_test.exe by entering `.\basic_test.exe` into the VS Code terminal.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "LAB-00" with files like testA.cpp, testB.cpp, basic_test.cpp, build.bat, gtest.bat, testA.exe, testB.exe, and README.md.
- Terminal:** The "TERMINAL" tab is selected, showing the output of a build process and a test run. The terminal command is:


```
PS C:\Users\mlehe\Desktop\lab-00\build> ./basic_test
```

 The output shows the compilation of basic_test.cpp and the execution of the test suite:


```
[=====] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from BASIC_TEST
[RUN]   1 test from BASIC_TEST
[OK]     OK | BASIC_TEST.BasicTest (0 ms)
[-----] 1 test from BASIC_TEST (0 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test suite ran. (1 ms total)
[PASSED] 1 test.
```
- Bottom Status Bar:** Shows "In 3, Col 89" and "Spaces: 4" and "UTF-8".

■ Getting started with the project ■

Find [basic_test.cpp](#)

You will be supplied with a `basic_test.cpp` file. You will copy this file and overwrite the existing *generic* `basic_test.cpp` in your project folder. After this, you will **never** edit the `basic_test.cpp` file.

`basic_test.cpp` demonstrates the functionality of the project and gives you an opportunity to make sure your function signatures and class declarations match those of the grader. You should be able to compile and run the `basic_test.cpp` with your functions.

Pay special attention to the `#include` path at the top. Your file structure has to be **exactly** the same as the one depicted here.

click [here](#) to download `basic_test.cpp` if you have not already.

```

File Edit Selection View Go Run Terminal Help
File Explorer Editor Terminal Help README.md basic_test.cpp ...
OPEN EDITORS
testA.cpp testB.cpp README.md basic_test.cpp ...
_ltests _test_files ...
basic_test.cpp 2, M
testA.cpp
testB.cpp
.github
build
googletest
basic_test.exe
build.bat
gtest.bat
testA.exe
testB.exe
cmake
includes
.gitignore
CMakeLists.txt
main.cpp
README.md
...
_ltests > _test_files > basic_test.cpp > ...
1 #include "gtest/gtest.h"
2 #include <iostream>
3 #include <iomanip>
4 #include "../..includes/array_functions/array_functions.h"
5 //-----
6 //----- COPY BASIC_TEST INTO THIS FILE.
7 //----- AND THEN,
8 //----- DO NOT EDIT THIS FILE ANY FURTHER
9 //-----
10
11 bool basic_test(bool debug=false){
12     const int MAX = 20;
13     int a[MAX];
14     int size = 5; //# of interesting elements in the array
15     _array_init(a, size, -1);
16     cout << "\n\nafter init: ";
17     _print_array(a, size);
18     cout << endl;
19     for (int i = size; i < 10; i++){
20         _append(a, size, i * 10);
21         _print_array(a, size);
22         cout << endl;
23     }
24     int index = _find(a, size, 70);
25     if (index >= 0){
26         cout << "found 70 at: " << _at(a, size, index) << endl;
27         cout << "changing 70 to 700: " << endl;
28         _at(a, size, index) = 700;
29         _print_array(a, size);
30         cout << endl;
31     }
32     cout << "\n\n";
33     return true;
34 }
35 TEST(BASIC_TEST, BasicTest) {
36     //EXPECT_EQ(1, <your individual test functions are called here>);
37 }
38 //EXPECT_EQ(1, <your individual test functions are called here>);

int main(int argc, char **argv) {
    ::testing::InitGoogleTest(&argc, argv);
    std::cout << "\n\n-----running basic_test.cpp-----\n\n";
    return RUN_ALL_TESTS();
}

```

Add a new folder to the `includes/` folder.

name this folder `array_functions`.

This is where you will add your `.h` and `.cpp` files

```

File Edit Selection View Go Run Terminal Help basic_test.cpp - lab-00 - Visual Studio Code
OPEN EDITORS
LAB-00
._tests > _test_files > basic_test.cpp > basic_test(bool)
1 #include "gtest/gtest.h"
2 #include <iostream>
3 #include <iomanip>
4 #include "../includes/array_functions/array_functions.h"
5 //-----COPY BASIC_TEST INTO THIS FILE.
6 //-----AND THEN,
7 //-----DO NOT EDIT THIS FILE ANY FURTHER
8 //-----
9 //-----10
11 bool basic_test(bool debug=false){
12     const int MAX = 20;
13     int a[MAX];
14     int size = 5; // # of interesting elements in the array
15     _array_init(a, size, -1);
16     cout << "\n\nafter init: ";
17     _print_array(a, size);
18     cout << endl;
19     for (int i = size; i < 10; i++){
20         _append(a, size, i * 10);
21         _print_array(a, size);
22         cout << endl;
23     }
24     int index = _find(a, size, 70);
25     if (index >= 0){
26         cout << "Found 70 at: " << _at(a, size, index) << endl;
27         cout << "changing 70 to 700: " << endl;
28         _at(a, size, index) = 700;
29         _print_array(a, size);
30         cout << endl;
31     }
32     cout << "\n\n";
33     return true;
34 }
35 TEST(BASIC_TEST, BasicTest) {
36     //EXPECT_EQ(1, <your individual test functions are called here>);
37 }
38

```

master CMake: [Debug] Ready No Kit Selected Build [all] Type here to search Ln 20, Col 6 Spaces: 2 UFT-8 CRLF C++ Go Live Win32 12:27 PM 2/6/2021

Add two files to this folder.

Name these two files `array_functions.h` and `array_functions.cpp`

```

File Edit Selection View Go Run Terminal Help basic_test.cpp - lab-00 - Visual Studio Code
OPEN EDITORS
LAB-00
._tests > _test_files > basic_test.cpp > array_functions.h > array_functions.cpp
1 #include "gtest/gtest.h"
2 #include <iostream>
3 #include <iomanip>
4 #include "../includes/array_functions/array_functions.h"
5 //-----COPY BASIC_TEST INTO THIS FILE.
6 //-----AND THEN,
7 //-----DO NOT EDIT THIS FILE ANY FURTHER
8 //-----
9 //-----10
10
11 bool basic_test(bool debug=false){
12     const int MAX = 20;
13     int a[MAX];
14     int size = 5; // # of interesting elements in the array
15     _array_init(a, size, -1);
16     cout << "\n\nafter init: ";
17     _print_array(a, size);
18     cout << endl;
19     for (int i = size; i < 10; i++){
20         _append(a, size, i * 10);
21         _print_array(a, size);
22         cout << endl;
23     }
24     int index = _find(a, size, 70);
25     if (index >= 0){
26         cout << "Found 70 at: " << _at(a, size, index) << endl;
27         cout << "changing 70 to 700: " << endl;
28         _at(a, size, index) = 700;
29         _print_array(a, size);
30         cout << endl;
31     }
32     cout << "\n\n";
33     return true;
34 }
35 TEST(BASIC_TEST, BasicTest) {
36     //EXPECT_EQ(1, <your individual test functions are called here>);
37 }
38

```

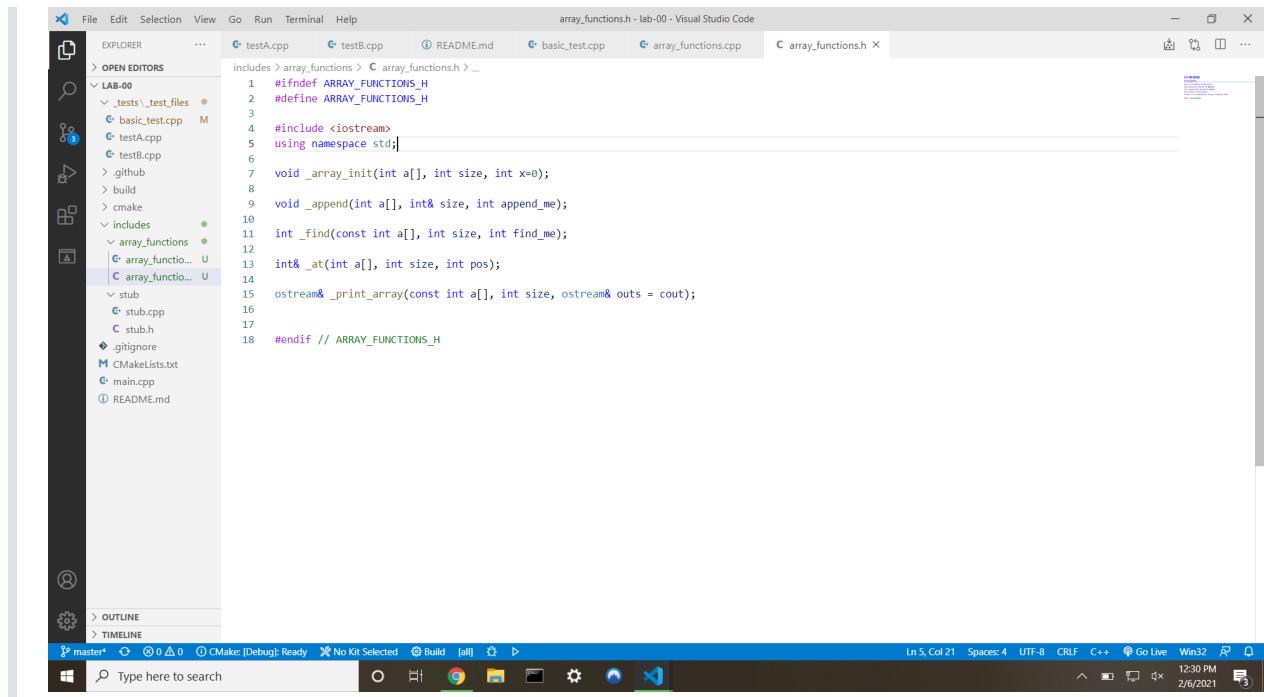
master CMake: [Debug] Ready No Kit Selected Build [all] Type here to search Ln 20, Col 6 Spaces: 2 UFT-8 CRLF C++ Go Live Win32 12:27 PM 2/6/2021

Add the function signatures.

add these function signatures to the `array_functions.h` file:

```
void _array_init(int a[], int size, int x=0);
void _append(int a[], int& size, int append_me);
int _find(const int a[], int size, int find_me);
int& _at(int a[], int size, int pos);
ostream& _print_array(const int a[], int size, ostream& outs = cout);
```

Normally, you will either be given these function signatures or you will *deduce* them from the code in `basic_test.cpp`



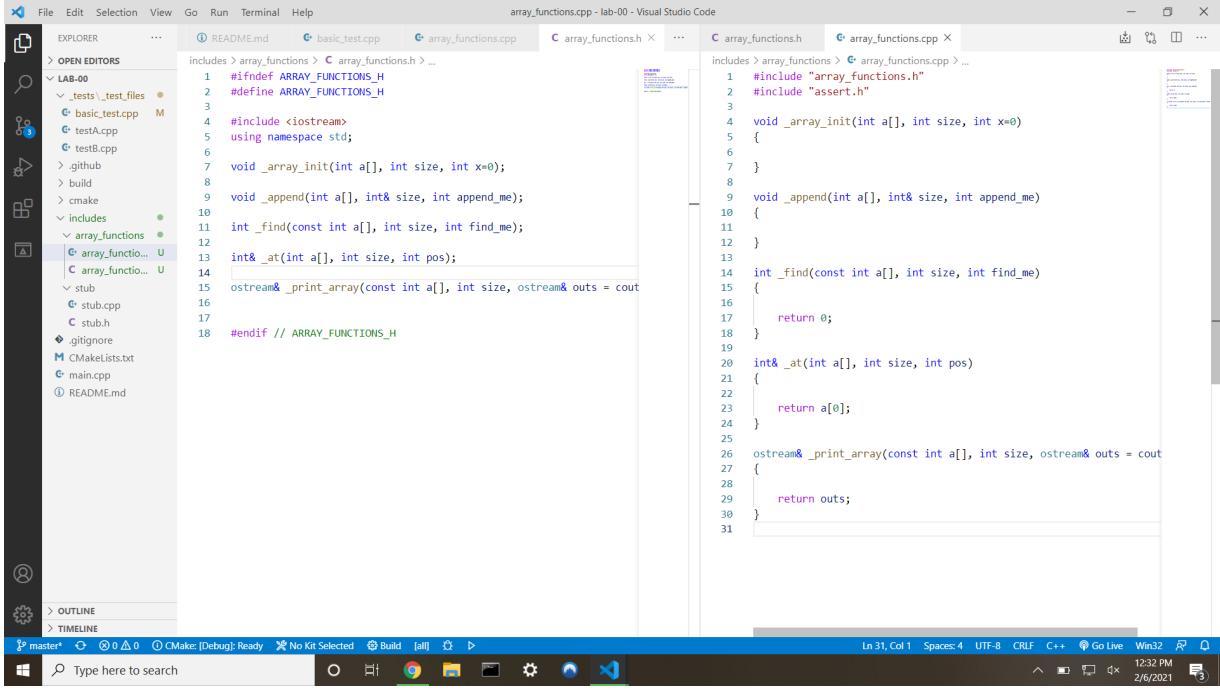
Write function *stubs*

Function stubs are just function signatures with a return statement if needed.

Function stubs are a quick way to get the project up and running. I find the students who adopt this method in their workflow have an easier time completing projects.

TIP:

I normally copy the function signatures and paste them into the `.cpp` file. Then, I replace the `;` at the end of the line with braces `({})`. Then, I add the returns whenever necessary.



```

array_functions.h
includes > array_functions > C array_functions.h > ...
1 #ifndef ARRAY_FUNCTIONS_H
2 #define ARRAY_FUNCTIONS_H
3
4 #include <iostream>
5 using namespace std;
6
7 void _array_init(int a[], int size, int x=0);
8
9 void _append(int a[], int& size, int append_me);
10
11 int _find(const int a[], int size, int find_me);
12
13 int& _at(int a[], int size, int pos);
14
15 ostream& _print_array(const int a[], int size, ostream& outs = cout
16
17 #endif // ARRAY_FUNCTIONS_H

array_functions.cpp
includes > array_functions > C array_functions.cpp > ...
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x=0)
5 {
6
7 }
8
9 void _append(int a[], int& size, int append_me)
10 {
11
12 }
13
14 int _find(const int a[], int size, int find_me)
15 {
16
17     return 0;
18 }
19
20 int& _at(int a[], int size, int pos)
21 {
22
23     return a[0];
24 }
25
26 ostream& _print_array(const int a[], int size, ostream& outs = cout
27
28
29 }
30
31

```

Build googletest framework

Go to `build/` and run `gtest.bat` :

`cd` into the `build/` folder, and from there, run `.\gtest.bat` Remember, you only do this per project. So, if you have already done this in the previous steps, you should skip it.

Now, we are ready to compile our project using `.\build.bat`

If you followed these steps faithfully, you will have the same syntax errors that I had, namely that all those functions we defined in `array_functions.h` and `.cpp` are **undefined!**

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like basic_test.cpp, array_functions.h, testA.cpp, testB.cpp, and README.md.
- Code Editor:** Displays the content of array_functions.h, which includes `#include <iostream>` and defines functions `_array_init` and `append`.
- Terminal:** Shows the output of the build process. It starts with `PS C:\Users\melhe\Desktop\lab-00> cd build`, followed by multiple error messages from `\build.bat` indicating undefined references to symbols defined in array_functions.h (e.g., `_array_init`, `append`, `print_array`, etc.).
- Status Bar:** Shows the current file is array_functions.h, the line count is 14, column 1, and the date/time is 2/6/2021 12:33 PM.

build.bat errors, zoomed in:

Here is a closer, more readable look at the errors reported by `.\build.bat`

```
PS C:\Users\melhe\Desktop\lab-00\build> .\build.bat
C:\Users\melhe\Desktop\lab-00\build>g++ -std=gnu++11 -o basic_test .._tests/_test_files/basic_test.cpp ..includes/stub/stub.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/build/lib -lgtest
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0x34): undefined reference to `__array_init(int*, int, int)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0x62): undefined reference to `__print_array(int const*, int, std::ostream&)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0xa4): undefined reference to `__append(int*, int, int)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0xbe): undefined reference to `__print_array(int const*, int, std::ostream&)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0xf2): undefined reference to `__find(int const*, int, int)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0x12e): undefined reference to `__at(int*, int, int)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0x18c): undefined reference to `__at(int*, int, int)'
c:/mingw/bin/..lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12qe9.o:basic_test.cpp:(.text+0x1ac): undefined reference to `__print_array(int const*, int, std::ostream&)'
collect2.exe: error: ld returned 1 exit status
```

the batch file

We will spare you the suspense. The reason for this error is that we never added `array_functions.cpp` to our `build.bat`.

A brief explanation of the line of code in build.bat follows:

```
g++ -std=gnu++11 -o basic_test ../_tests/_test_files/basic_test.cpp
..../includes/stub/stub.cpp -Igoogletest/googletest/include
```

`basic_test` in this command is the name of the executable that will be created.

Starting from `-Igoogletest/googletest/include`, there is nothing you need to change in the line. This is a googletest option.

All .cpp files must be listed in the build.bat between the executable name and the googletest options at the end of each command.

```
g++ -std=gnu++11 -o basic_test ../_tests/_test_files/basic_test.cpp ..../includes/stub/stub.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/googletest/include -lpthread -Lgoogletest/googletest/build/lib -lgtest
```

```
PS C:\Users\melhe\Desktop\lab-00\build> ./build.bat
C:\Users\melhe\Desktop\lab-00\build>g++ -std=gnu++11 -o basic_test ../_tests/_test_files/basic_test.cpp ..../includes/stub/stub.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/googletest/include -lpthread -Lgoogletest/googletest/build/lib -lgtest
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x34): undefined reference to `__array_in
l((int*, int, int))
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x62): undefined reference to `__print_ar
ray(int const*, int, std::ostream&)
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x64): undefined reference to `__appendin
nt*, int*, int*)
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x6e): undefined reference to `__print_ar
ray(int const*, int, std::ostream&)
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0xf2): undefined reference to `__findint
const*, int, int)
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x12e): undefined reference to `__at(int*
, int, int)
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x18c): undefined reference to `__at(int*
, int, int)
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:\Users\melhe\AppData\Local\Temp\ccq12Qe9.o:basic_test.cpp:(.text+0x1ac): undefined reference to `__print_ar
ray(int const*, int, std::ostream&)
collect2.exe: error: ld returned 1 exit status
C:\Users\melhe\Desktop\lab-00\build>g++ -std=gnu++11 -o testA ../_tests/_test_files/testA.cpp ..../includes/stub/stub.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/googletest/include -lpthread -Lgoogletest/googletest/build/lib -lgtest
```

Editing the batch file

```
g++ -std=gnu++11 -o basic_test ../_tests/_test_files/basic_test.cpp ..../includes/array_functions/array_functions.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/googletest/include -lgtest
g++ -std=gnu++11 -o testA ../_tests/_test_files/testA.cpp ..../includes/stub/stub.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/googletest/build/lib -lgtest
g++ -std=gnu++11 -o testB ../_tests/_test_files/testB.cpp ..../includes/stub/stub.cpp -Igoogletest/googletest/include -pthread -Lgoogletest/googletest/build/lib -lgtest
```

back at the `CMakeLists.txt`:

Even you, who uses windows and may not have any use for the `CMakeLists.txt` *locally*, **MUST** make sure `CMakeLists.txt` reflects the current state of your project. Otherwise, the grader will not be able to compile and run your project.

Notice that we are missing the `array_functions.cpp` from the `basic_test`
`ADD_EXECUTABLE` statement:

So, let's add it...

File Edit Selection View Go Run Terminal Help CMakeLists.txt - lab-00 - Visual Studio Code

EXPLORER **OPEN EDITORS**

LAB-00

- tests_test_files
- basic_test.cpp M
- testA.cpp
- testB.cpp
- .github
- build
- googletest
- build.bat M
- gtestbat
- testAExe M
- testBExe M
- cmake
- includes
- .gitignore
- CMakeLists.txt M
- main.cpp
- README.md

CMakeLists.txt

```
31
32 ADD_EXECUTABLE(main
33     main.cpp
34     includes/stub/stub.cpp
35 )
36
37 ADD_EXECUTABLE(basic_test
38     _tests/_test_files/basic_test.cpp
39     includes/array_functions/array_functions.cpp
40 )
41 ADD_EXECUTABLE(testA
42     _tests/_test_files/testA.cpp
43     includes/stub/stub.cpp
44 )
45
46 ADD_EXECUTABLE(testB
47     _tests/_test_files/testB.cpp
48     includes/stub/stub.cpp
49 )
50
51 TARGET_LINK_LIBRARIES(basic_test gtest)
52 TARGET_LINK_LIBRARIES(testA gtest)
53 TARGET_LINK_LIBRARIES(testB gtest)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: powershell

```
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:/Users/melhe/AppData/Local/Temp/ccq12qe9.o:basic_test.cpp:(.text+0x12e): undefined reference to `__at(int*, int, int*)'
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:/Users/melhe/AppData/Local/Temp/ccq12qe9.o:basic_test.cpp:(.text+0x18c): undefined reference to `__at(int*, int, int*)'
c:/mingw/bin/../lib/gcc/mingw32/9.2.0/../../../../mingw32/bin/ld.exe: C:/Users/melhe/AppData/Local/Temp/ccq12qe9.o:basic_test.cpp:(.text+0x1ac): undefined reference to `__print_array<const*, int, std::ostream&*>'
collect2.exe: error: ld returned 1 exit status

C:/Users/melhe/Desktop/lab-00/buildg++ -std=gnu++11 -o testA .._tests/_test_files/testA.cpp ..includes/stub/stub.cpp -Igoogletest/include -pthread -Igoogletest/buil
ld/lib -lgtest

C:/Users/melhe/Desktop/lab-00/buildg++ -std=gnu++11 -o testB .._tests/_test_files/testB.cpp ..includes/stub/stub.cpp -Igoogletest/include -pthread -Igoogletest/buil
ld/lib -lgtest

PS C:/Users/melhe/Desktop/lab-00/build
```

mster* File Edit Selection View Go Run Terminal Help CMakeLists.txt - lab-00 - Visual Studio Code

0 0 △ 0 CMake [Debug] Ready No Kit Selected Build [all] ↻ ▶

In 39, Col 49 [44 selected] Spaces: 4 UTF-8 CR/LF Plain Text Go Live

12:39 PM 2/6/2021

```
Add array_functions.cpp to the  
ADD_EXECUTABLE(basic_test... )
```

Do not use commas to separate the files.

Do NOT include .h files.

Normally, **all** three executables will need all the .cpp files

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `testA.cpp`, `testB.cpp`, `README.md`, `basic_test.cpp`, `array_functions.cpp`, `array_functions.h`, `build.bat`, and `CMakeLists.txt`.
- Code Editor:** The `CMakeLists.txt` file is open, displaying CMake configuration code. It includes sections for adding executables (`ADD_EXECUTABLE`) and target linking (`TARGET_LINK_LIBRARIES`). Lines 48-53 show the linking of `gtest` for `testB`. Error messages from `collect2.exe` are visible at the bottom of the editor.
- Terminal:** The terminal tab shows command-line output for building the project. It includes commands like `mingw32/bin/g++` and `ld` for linking, and `g++` for compilation. It also shows the creation of executables `testA` and `testB`.
- Status Bar:** The status bar at the bottom indicates the current branch is `master`, the build configuration is `Debug`, and there are no errors or warnings. It also shows the file count as 11, the current line as 48, and the column as 49.

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like testA.cpp, testB.cpp, README.md, basic_test.cpp, array_functions.cpp, array_functions.h, build.bat, and CMakeLists.txt.
- Code Editor:** The CMakeLists.txt file is open, displaying CMake configuration code for building executables testA and testB, and a library basic_test. It includes targets for gtest and pthread.
- Terminal:** A powershell terminal window is open at the bottom, showing the command `collect2.exe: error: ld returned 1 exit status` and the full command used to run the build.
- Status Bar:** Shows the current branch as "master", the build configuration as "Debug", and the message "CMake: [Debug] Ready".

.\build.bat again:

Let's run .\build again and pray that...

... and, we have more syntax errors. Default arguments can only be specified in the declaration of the function and **not** in the definition.

So, we must remove all those default values for the defalut arguments on every function.

The screenshot shows the Visual Studio Code interface with the array_functions.cpp file open. The terminal window displays several error messages from the build process:

```

PS C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o basic_test ..\_tests\_test_files/basic_test.cpp ..\includes\array_functions\array_functions.cpp -Igoogletest\gtest
..\includes\array_functions\array_functions.cpp:4:6: error: default argument given for parameter 3 of 'void _array_init(int*, int, int)' [-fpermissive]
  4 | void _array_init(int a[], int size, int x=0);
     |           ~~~~~^~~~~~
In file included from ..\includes\array_functions\array_functions.cpp:1:
..\includes\array_functions\array_functions.h:7:6: note: previous specification in 'void _array_init(int*, int, int)' here
  7 | void _array_init(int a[], int size, int x=0);
     |           ~~~~~^~~~~~
..\includes\array_functions\array_functions.cpp:26:10: error: default argument given for parameter 3 of 'std::ostream& _print_array(const int*, int, std::ostream&)' [-fpermissive
  26 | ostream& _print_array(const int a[], int size, ostream& outs = cout)
     |           ~~~~~^~~~~~
In file included from ..\includes\array_functions\array_functions.cpp:1:
..\includes\array_functions\array_functions.h:15:10: note: previous specification in 'std::ostream& _print_array(const int*, int, std::ostream&)' here
  15 | ostream& _print_array(const int a[], int size, ostream& outs = cout)
     |           ~~~~~^~~~~~

```

The code editor shows the implementation of the `_array_init` and `_print_array` functions.

Fix the `_print_array` function...

The screenshot shows the Visual Studio Code interface with the array_functions.cpp file open. The terminal window displays the same error messages as before, indicating that the fix has not yet been applied to the `_print_array` function.

by removing the default value = cout

Same with `_array_init`:

```

array_functions.h
includes > array_functions > C array_functions.h > ...
1 #ifndef ARRAY_FUNCTIONS_H
2 #define ARRAY_FUNCTIONS_H
3
4 #include <iostream>
5 using namespace std;
6
7 void _array_init(int a[], int size, int x=0);
8
9 void _append(int a[], int& size, int append_me);
10
11 int _find(const int a[], int size, int find_me);
12
13 int& _at(int a[], int size, int pos);
14
15 ostream& _print_array(const int a[], int size, ostream& outs = cout)
16
17 #endif // ARRAY_FUNCTIONS_H

array_functions.cpp
includes > array_functions > C array_functions.cpp > _array_init(int [], int, int)
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x)
5 {
6
7 }
8
9 void _append(int a[], int& size, int append_me)
10 {
11
12 }
13
14 int _find(const int a[], int size, int find_me)
15 {
16
17     return 0;
18 }

array_functions.cpp
includes > array_functions > C array_functions.cpp > _array_init(int [], int, int)
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x)
5 {
6
7 }
8
9 void _append(int a[], int& size, int append_me)
10 {
11
12 }
13
14 int _find(const int a[], int size, int find_me)
15 {
16
17     return 0;
18 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\melhe\Desktop\lab-00\build> ./build.bat
C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o basic_test .._tests/_test_files/basic_test.cpp ..\includes\array_functions\array_functions.cpp -Igtest -Igtest/include -pthread -Lgtest -Lgtest/include -lgtest
C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o testA .._tests/_test_files/testA.cpp ..\includes\stub\stub.cpp -Igtest -Igtest/include -pthread -Lgtest -Lgtest/include -lgtest
C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o testB .._tests/_test_files/testB.cpp ..\includes\stub\stub.cpp -Igtest -Igtest/include -pthread -Lgtest -Lgtest/include -lgtest
PS C:\Users\melhe\Desktop\lab-00\build>

```

`.\build.bat` one more time:

and this time it will run successfully.

This is a huge step. We now have a working project even though our functions are basically empty.

You can even run `basic_test.exe`. Of course this will not run satisfactorily. You will get mostly garbage. -afterall, we are running on stubs!- but it **does** run!!

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "LAB-00" with files like "array_functions.cpp", "array_functions.h", "build.bat", "CMakeLists.txt", "main.cpp", and "README.md".
- Editor:** Two tabs are open: "array_functions.cpp" and "array_functions.h". The code in array_functions.h includes "#include <iostream>" and "using namespace std;". The code in array_functions.cpp includes "#include <array>" and "#include <assert.h>".
- Terminal:** The terminal shows the command "ld/lib -lgtest" being run, followed by the path "PS C:\Users\melhe\Desktop\lab-00\build> .\basic_test.exe". Below this, the output of the test execution is shown, indicating 1 test from BASIC_TEST was run successfully.
- Bottom Status Bar:** Shows the current file is "array_functions.cpp", line 7, column 2, with 4 spaces, and the status "CMake: [Debug] Ready".

run `git status` , add , and commit with the message success on make with stubs

The importance of having regular `commit` s in your project cannot be overstated. This is a large part of the evaluation of your project by me.

```

array_functions.cpp - lab-00 - Visual Studio Code
array_functions.h
array_functions.cpp
CMakeLists.txt

array_functions.h ...
includes > array_functions > C array_functions.h ...
1 #ifndef ARRAY_FUNCTIONS_H
2 #define ARRAY_FUNCTIONS_H
3
4 #include <iostream>
5 using namespace std;
6
7 void _array_init(int a[], int size, int x=0);
8
9 void _append(int a[], int& size, int append_me);
10
11 int _find(const int a[], int size, int find_me);
12
13 int& _at(int a[], int size, int pos);
14
15 ostream& _print_array(const int a[], int size, ostream& outs = cout)
16
17 {
18     return outs;
19 }
20
21 #endif // ARRAY_FUNCTIONS_H

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\melihe\Desktop\lab-00\build> git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: ..\gitignore
modified: ..\CMakeLists.txt
modified: ..\tests_\test_files\basic_test.cpp
modified: build.bat

Untracked files:
(use "git add <file>..." to include in what will be committed)
..\.includes\array_functions/

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\melihe\Desktop\lab-00\build> [REDACTED]

Ln 9, Col 4 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 106 PM 2/6/2021

```

array_functions.cpp - lab-00 - Visual Studio Code
array_functions.h
array_functions.cpp
CMakeLists.txt

array_functions.h ...
includes > array_functions > C array_functions.h ...
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x)
5 {
6
7 }
8
9 void _append(int a[], int& size, int append_me)
10 {
11
12 }
13
14 int _find(const int a[], int size, int find_me)
15 {
16
17 }
18
19 int& _at(int a[], int size, int pos);
20
21

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: ..\gitignore
modified: ..\CMakeLists.txt
modified: ..\tests_\test_files\basic_test.cpp
modified: build.bat

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: ..\gitignore
modified: ..\CMakeLists.txt
modified: ..\tests_\test_files\basic_test.cpp
modified: build.bat

Untracked files:
(use "git add <file>..." to include in what will be committed)
..\.includes\array_functions/

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\melihe\Desktop\lab-00\build> git add .
PS C:\Users\melihe\Desktop\lab-00\build> git commit -m "success running with stubs"
[master 7cd7f0d] success running with stubs
1 file changed, 2 insertions(+), 2 deletions(-)
PS C:\Users\melihe\Desktop\lab-00\build> [REDACTED]

Ln 9, Col 4 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 108 PM 2/6/2021

**Implement `_array_init` and `_print_array`.
`testB.cpp` can be seen waiting to host the test functions.**

Now, we can go in and implement the functions one by one and write tests for them.
These tests will be written in the `testB.cpp` file.

■ Writing Tests: ■

testB : Our first test:

After implementing the `_array_init` and `_print_array` functions, we will write a simple test that will verify that the `_init` function works as it should.

The test function is boolean. It returns `true` if the init function works properly and false otherwise.

Call the `_array_init` function and then go through each and every cell and verify that each element is `-1`.

If you find one cell that is not `-1`, return `false`.

Note also that we return `true` at the end of the test function. I do this in every test function I write.

The TEST function:

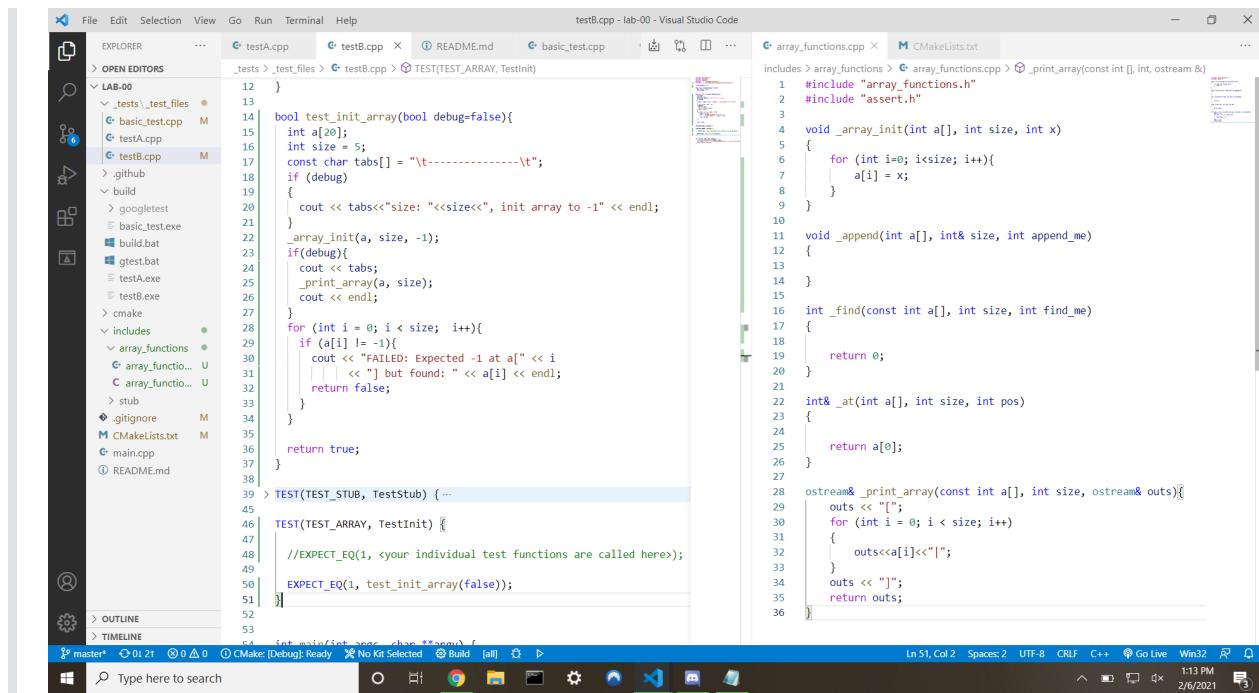
The `TEST` function is part of the googletest testing framework. To simplify our work, we always use the same format for the `TEST` function: Declare a `bool success` and assign it to the return value of the test function.

Then, compare `success` with `1` or `true`

A quick word about the two arguments of the `TEST` function:

The first is the name of the *test suit* and the second is the name of this very test. Each test suit may contain multiple tests. Later, we will write another test for the `_append` function with the same first argument as this test: `TEST_ARRAY`. By the time we are done, the `TEST_ARRAY` test suite will have three individual tests.

Pay attention to the **naming conventions** for this course: The test suite will be in ALL CAPS with underscores between the words. The test names will be camel case and regular function names are all lower case with underscores.



```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS testA.cpp testB.cpp README.md basic_test.cpp
LAB-00 _tests/_test_files basic_test.cpp M testA.cpp testB.cpp M
github build googletest basic_test.exe build.bat gtest.bat testA.exe testB.exe cmake includes array_functions
array_functions.h array_functions.cpp M CMakeLists.txt M main.cpp README.md
array_functions.cpp
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x)
5 {
6     for (int i=0; i<size; i++){
7         a[i] = x;
8     }
9 }
10 void _append(int a[], int size, int append_me)
11 {
12 }
13
14 int _find(const int a[], int size, int find_me)
15 {
16     return 0;
17 }
18
19 int& _at(int a[], int size, int pos)
20 {
21     return a[0];
22 }
23
24 ostream& _print_array(const int a[], int size, ostream& outs){
25     outs << "[";
26     for (int i = 0; i < size; i++) {
27         outs << a[i]<< "|";
28     }
29     outs << "]";
30     return outs;
31 }
32
33 EXPECT_EQ(1, _at(a, size, 0));
34
35 EXPECT_EQ(1, _find(a, size, 0));
36
37 }
38
39 > TEST(TEST_STUB, TestStub) { ...
40
41     TEST(TEST_ARRAY, TestInit) {
42
43         //EXPECT_EQ(1, <your individual test functions are called here>);
44
45         EXPECT_EQ(1, _init_array(false));
46     }
47 }
48
49
50
51
52
53

```

build and RUN!!

This time, we will run `.\build.bat` successfully and then, we run the `testB` executable by typing `.\testB.exe`

This will display two successful test runs: one for the `stub` test that was already part of the project, and one for the `TestInit` that we just wrote.

This means that our test function returned true .

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files and folders related to the project, including `basic_test.cpp`, `testA.cpp`, `testB.cpp`, `array_functions.h`, and `array_functions.cpp`. The `array_functions.cpp` editor shows implementation details for `_array_init` and `_find` functions. The terminal window at the bottom displays the build process and logs for `array_functions.cpp` and `testB.cpp`.

```

array_functions.cpp
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x)
5 {
6     for (int i=0; i<size; i++){
7         a[i] = x;
8     }
9 }
10
11 void _append(int a[], int& size, int append_me)
12 {
13 }
14
15 int _find(const int a[], int size, int find_me)
16 {
17     return 0;
18 }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

```

```

PS C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o basic_test ../_tests/_test_files/basic_test.cpp ..includes/array_functions/array_functions.cpp -Igoogletest/include -pthread -Lgoogletest/build/lib -lgtest
C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o testA ../_tests/_test_files/testA.cpp ..includes/stub/stub.cpp -Igoogletest/include -pthread -Lgoogletest/build/lib -lgtest
C:\Users\melhe\Desktop\lab-00\build> g++ -std=gnu++11 -o testB ../_tests/_test_files/testB.cpp ..includes/stub/stub.cpp -Igoogletest/include -pthread -Lgoogletest/build/lib -lgtest
PS C:\Users\melhe\Desktop\lab-00\build>

```

The screenshot shows the Visual Studio Code interface with the terminal window active. The terminal output shows the build of `array_functions.cpp` and `testB.cpp`, followed by the execution of `testB`. The test results indicate 2 tests passed.

```

array_functions.cpp
1 #include "array_functions.h"
2 #include "assert.h"
3
4 void _array_init(int a[], int size, int x)
5 {
6     for (int i=0; i<size; i++){
7         a[i] = x;
8     }
9 }

-----running testB.cpp-----
[=====] Running 2 tests from 2 test suites.
[=====] Global test environment set-up.
[=====] 1 test from TEST_STUB
[RUN]   TEST_STUB.TestsStub
[OK]   TEST_STUB.TestsStub (0 ms)
[=====] 1 test from TEST_ARRAY
[RUN]   TEST_ARRAY.TestInit
[OK]   TEST_ARRAY.TestInit (0 ms)
[=====] 1 test from TEST_ARRAY (3 ms total)

[=====] Global test environment tear-down
[=====] 2 tests from 2 test suites ran. (56 ms total)
[PASSED] 2 tests.

PS C:\Users\melhe\Desktop\lab-00\build>

```

Implement `_append` and `_at` :

We have thus far implemented `_array_init` and `print_array`. Let's implement `_append` and `_at` as well.

You will *borrow* my code for this particular lab, but make sure you comment the code very well.

Once we have implemented `_append`, write the test for it in `testB.cpp`. Don't forget to add a `TEST()` for the `test_append()` function.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "LAB-00". Files include `array_functions.cpp`, `basic_test.cpp`, `testA.cpp`, `testB.cpp`, `README.md`, `CMakeLists.txt`, and several build scripts and configuration files.
- Code Editor:** The right pane displays the `array_functions.cpp` file. The code implements array manipulation functions like `_array_init`, `_append`, `_at`, and `_print_array`. It includes assertions and handles size validation.
- Status Bar:** Shows the current file is `array_functions.cpp`, the line number is 27, column 1, and the status bar also includes information about spaces, encoding, and file modification time (2/6/2021, 12:22 PM).

Obviously, this is done in the `testB.cpp` file. Again, do not forget to add the `TEST()` function for `test_at()`

Once again, you will *borrow* my code for this particular lab, but make sure you comment the code very well.

The screenshot shows a Visual Studio Code window with the following details:

- File Explorer (Left):** Shows the project structure with files like `array_functions.cpp`, `basic_test.cpp`, `testA.cpp`, and `testB.cpp`.
- Code Editor (Top Right):** Displays the `array_functions.cpp` file content, which includes functions for initializing arrays, appending values, finding elements, and printing arrays.
- Code Editor (Bottom Right):** Displays the `testB.cpp` file content, which contains a test function for the `_array_init` function.
- Status Bar:** Shows the current branch as "master", the commit hash "012f", and the status "CMake: [Debug]: Ready".
- Taskbar (Bottom):** Includes icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and various system icons.

`.\build.bat` and run `testB.cpp` again:

Let's .\build.bat and run testB to make sure our test_append and test_at pass:

File Edit Selection View Go Run Terminal Help

testB.cpp - lab-00 - Visual Studio Code

EXPLORER OPEN EDITORS

LAB-00

_tests _test_files

basic_test.cpp M

testA.cpp M

testB.cpp M

.github M

build

googletest

basic_testexe

build.bat

gtest.bat

testA.exe

testB.exe

cmake

includes

array_functions

array_function.h U

array_function.cpp U

gitignore M

CMakeLists.txt M

main.cpp

README.md

testB.cpp

TEST(TEST_ARRAY, TestInit) { ...
TEST(TEST_ARRAY, TestAppend) {
//EXPECT_EQ(1, <your individual test functions are called here>);
EXPECT_EQ(1, test_append(false));
TEST(TEST_ARRAY, TestAT) {
//EXPECT_EQ(1, <your individual test functions are called here>);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\meihe\Desktop\lab-00\build> ./testB.exe

-----running testB.cpp-----

[=====] Running 4 tests from 2 test suites.
[=====] Global test environment set-up.
[=====] 1 test from TEST_STUB
[RUN] TEST_STUB.TestStub
[OK] TEST_STUB.TestStub (0 ms)
[=====] 1 test from TEST_STUB (2 ms total)

[=====] 3 tests from TEST_ARRAY
[RUN] TEST_ARRAY.TestInit
[OK] TEST_ARRAY.TestInit (0 ms)
[RUN] TEST_ARRAY.TestAppend
[OK] TEST_ARRAY.TestAppend (0 ms)
[RUN] TEST_ARRAY.TestAT
[OK] TEST_ARRAY.TestAT (0 ms)
[=====] 3 tests from TEST_ARRAY (70 ms total)

[=====] Global test environment tear-down
[=====] 4 tests from 2 test suites ran. (80 ms total)
[PASSED] 4 tests.

PS C:\Users\meihe\Desktop\lab-00\build>

array_functions.cpp

array_functions.h

assert.h

_array_init(int a[], int size, int x)

for (int i=0; i<size; i++)
a[i] = x;

_append(int a[], int& size, int append_me)

afsize+=1 = append me;

Ln 126, Col 1 Spaces: 2 UTF-8 CRLF C++ Go Live Win32

master 0:121 0:0:0 CMake: [Debug]: Ready No Kit Selected Build [all]

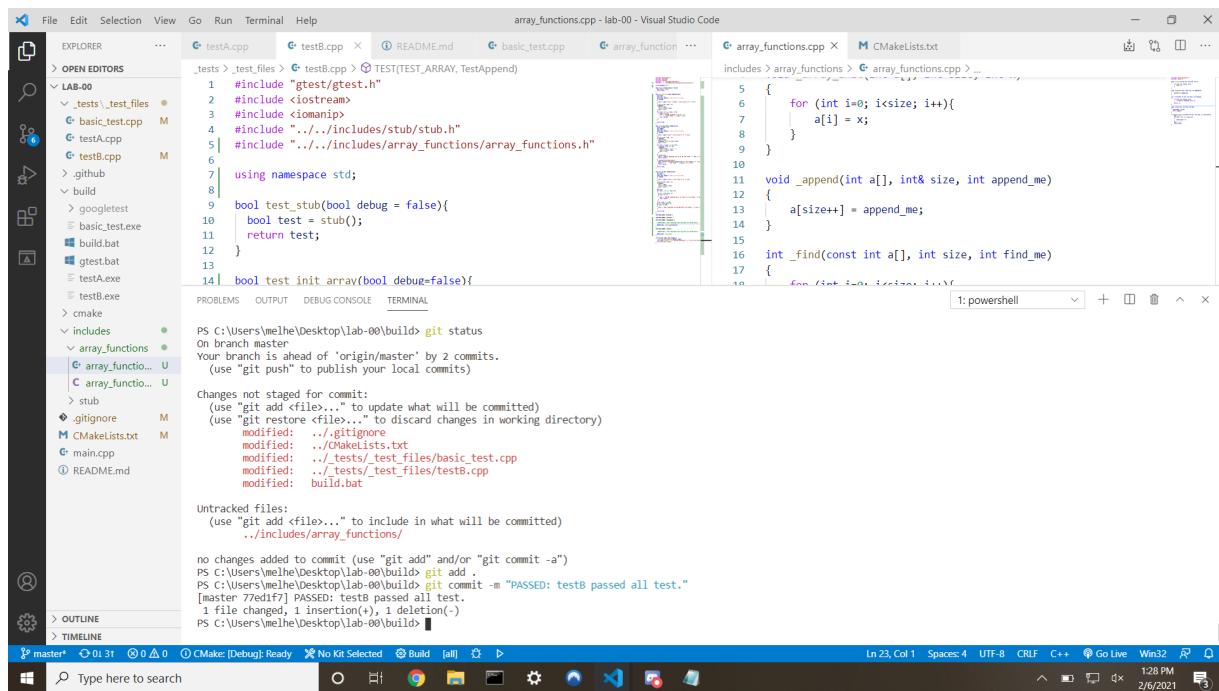
Type here to search

■ Completing the project ■

Implement the `_find()` function on your own

You will also write a `test_find()` function. Once you have implemented `_find()` and written the test function for it (don't forget to comment) you are ready to `.\build.bat` and run `testB` once again.

Once you have successfully run `testB` with `_find`, you `git add` and `commit` your changes. Once again, do not forget to update your `CMakeLists.txt` file for the autograder.



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows files in the project structure, including `testA.cpp`, `testB.cpp`, `README.md`, `basic_test.cpp`, `array_functions.cpp`, and `CMakeLists.txt`.
- Code Editor:** Displays the `array_functions.cpp` file with the following code:

```

1 #include "gtest/gtest.h"
2 #include <iostream>
3 #include <iomanip>
4 #include "../../.includes/stub/stub.h"
5 #include "../../.includes/array_functions/array_functions.h"
6
7 using namespace std;
8
9 bool test_stub(bool debug = false){
10     bool test = stub();
11     return test;
12 }
13
14 bool test_init_array(bool debug=false)

```
- Terminal:** Shows a powershell terminal window with the following output:

```

PS C:\Users\melhe\Desktop\lab-00\build> git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
        modified: ..\gitignore
        modified: ..\CMakeLists.txt
        modified: ..\main.cpp
        modified: ..\README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ..\includes\array_functions

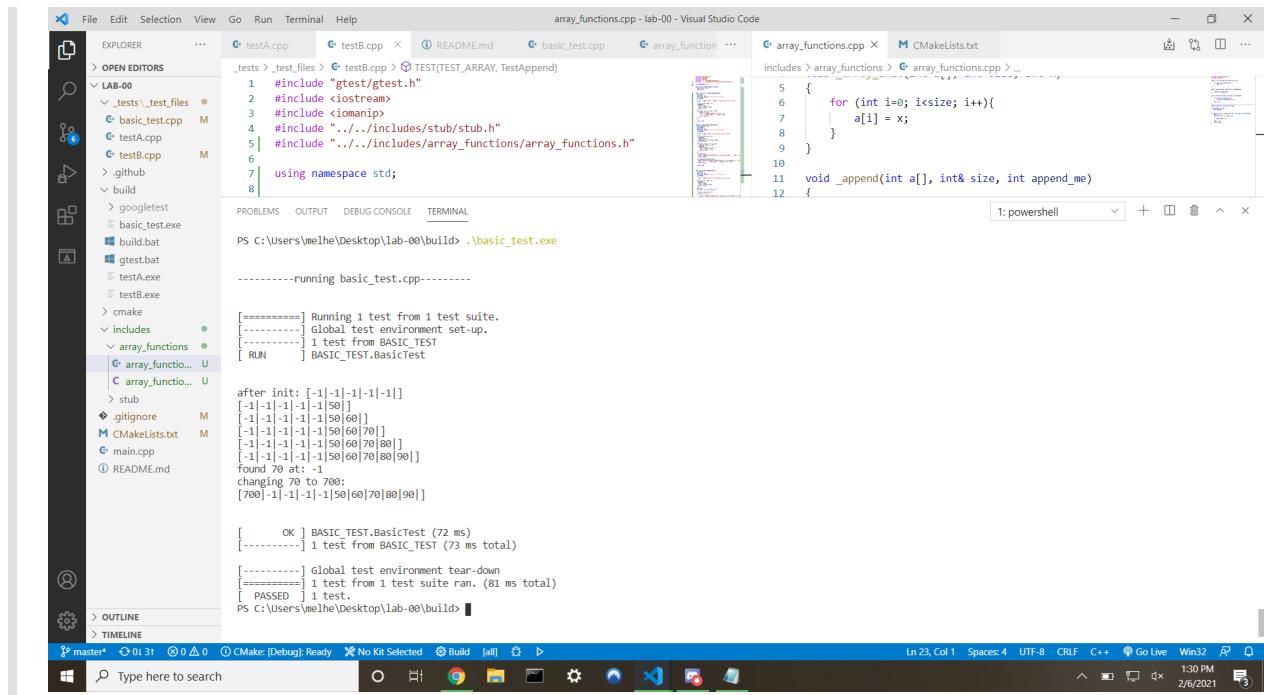
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\melhe\Desktop\lab-00\build> git add .
PS C:\Users\melhe\Desktop\lab-00\build> git commit -m "PASSED: testB passed all test."
[master 77edbf7] PASSED: testB passed all test.
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\melhe\Desktop\lab-00\build>

```

Finally, we can run `basic_test.cpp` :

Now that we have implemented all the functions that are used in `basic_test.cpp`, we can make and run this file.

I cannot overemphasize how important it is for this test to be able to compile and run **without** your editing it in any way. If your project cannot compile and run `basic_test`, the grader will not be able to run your project.



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under `LAB-00`. It includes files like `testA.cpp`, `testB.cpp`, `README.md`, `basic_test.cpp`, `array_functions.cpp`, and `CMakeLists.txt`.
- Code Editors:** There are two code editors open:
 - `array_functions.cpp`: Contains C++ code for an array manipulation function.
 - `array_function`: A CMake configuration file.
- Terminal:** Shows the command `PS C:\Users\melhe\Desktop\lab-00\build> ./basic_test.exe` being run, followed by the output of the test execution.
- Output:** Shows the results of the test, including the number of tests run, the time taken, and the status of each test case.

git add and the final git commit

Let's go back to the root directory by typing `cd ..` - remember that `..` means parent directory. `cd ..` means change directory to the parent.

My commit message will let me know what stage of the development I am in. I have just PASSED both the `basic_test` and `testB`

The screenshot shows the Visual Studio Code interface with several files open:

- testA.cpp**: Contains a test stub function.
- testB.cpp**: Contains a TEST macro definition.
- README.md**: A markdown file.
- array_functions.h**: An empty header file.
- array_functions.cpp**: Contains implementation for `_array_init`, `_append`, and `find`.
- CMakeLists.txt**: Configuration file for the project.

The terminal window shows the following output:

```
PS C:\Users\melhe\Desktop\lab-00> git add .
PS C:\Users\melhe\Desktop\lab-00> git commit -m "PASSED: testB passed all tests."
[master 7238ab8] PASSED: testB passed all tests.
 6 files changed, 273 insertions(+), 13 deletions(-)
create mode 100644 includes/array_functions/array_functions.h
create mode 100644 includes/array_functions/array_functions.cpp
PS C:\Users\melhe\Desktop\lab-00> git push origin master
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 2.75 KiB | 469.00 KiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To https://github.com/CS3A-classroom/lab_00-melh23.git
 7ed1f7..7238ab8 master -> master
PS C:\Users\melhe\Desktop\lab-00>
```

The status bar at the bottom right shows the file is ready (**CMake [Debug]: Ready**).

Autograder Status

You can keep track of the grading status of your project on the assignment page. A yellow dot means that the tests are still being compiled and run by the autograder. This shouldn't take more than a minute or two. You can refresh the page to update the status. A green checkmark means that all your tests have passed. A red x means that at least one test failed.

The screenshot shows a GitHub repository page for 'CS3A-classroom / lab_00-melh23'. The repository is private and was generated from 'CS3A-classroom/template_00_lab_0'. The 'Code' tab is selected, showing the 'master' branch with 1 branch and 0 tags. A commit by 'melh23' titled 'PASSED: testB passed all tests.' is shown, made 28 seconds ago with 9 commits. The commit details show a GitHub Classroom Autograding Workflow that passed all tests. The repository has 2 stars and 0 forks.

This screenshot shows the same GitHub repository page after a build has started. The commit log now includes two entries under the GitHub Classroom Workflow: 'CMake / build (push) ...' and 'GitHub Classroom Wo...', both of which are currently 'queued checks'. The rest of the repository details remain the same, including the star and fork counts.

To see a more in depth output of the autograder test runs, click on the Details link:

The screenshot shows a GitHub repository page for 'CS3A-classroom / lab_00-melh23'. The repository is private and was generated from 'CS3A-classroom/template_00_lab_0'. The code tab is selected, showing a commit by 'melh23' titled 'PASSED: testB passed all tests.' with a timestamp of '7238ab8 1 minute ago'. The commit message includes 'All checks have passed' and two successful checks: 'CMake / build (push)' and 'GitHub Classroom Wo...'. The repository has 9 commits in total. On the right side, there are sections for 'About', 'Releases', 'Packages', and 'Languages', each with a 'No releases published' or 'No packages published' message. The bottom of the screen shows a Windows taskbar with various icons.

It will take you to this page:

The screenshot shows a GitHub Actions run page for 'CS3A-classroom / lab_00-melh23/runs/1846556585'. The 'Actions' tab is selected, showing a single job named 'GitHub Classroom Workflow' that succeeded 3 minutes ago. The job details show an 'Autograding' step with a CMake action. A large portion of the page is a terminal-like log window titled 'Autograding' which shows the execution of a CMake-based autograding script. The log output includes several lines of CMake command-line output, including file paths like 'basic_test.cpp', and some internal logic related to test cases and global environment setup. The log ends with a successful test run and a final status message.

The screenshot shows a Windows desktop environment with a Microsoft Edge browser window open. The browser displays a GitHub pull request page for a repository named 'CS3A-classroom/lab_00-melh23'. The pull request has passed all tests, as indicated by the green checkmark icon and the message 'PASSED: testB passed all tests.' The pull request number is 7238ab8. The 'Actions' tab is selected, showing the 'Autograde' workflow. The 'Autograde' step has succeeded 7 minutes ago in 34s. The log output shows the execution of 'Run education/autograding@v1' and the results of the 'testB' test, which passed all 4 tests. The log concludes with 'All tests passed' and a series of colorful icons. The GitHub interface includes a sidebar for 'GitHub Classroom Workflow' and a 'Code' section. The taskbar at the bottom shows various pinned applications like iLearn, CS 164, School, CS, shit to read, research, google apps, GitHub, git stuff, lewis, and discord.

The screenshot shows a GitHub repository page for 'CS3A-classroom / lab_00-melh23'. The repository is private and was generated from 'CS3A-classroom/template_00_lab_0'. The 'Code' tab is selected. The repository has 2 pull requests, 0 stars, and 0 forks.

Code Issues Pull requests Actions Projects Security Insights ...

Code

melh23 added .bat files 6 minutes ago 4

.github GitHub Classroom Autograding Workflow 21 hours ago

_tests/_test_files Initial commit 21 hours ago

build added .bat files 6 minutes ago

cmake Initial commit 21 hours ago

includes/stub Initial commit 21 hours ago

.gitignore Initial commit 21 hours ago

CMakeLists.txt Initial commit 21 hours ago

README.md Initial commit 21 hours ago

main.cpp Initial commit 21 hours ago

README.md

00_LAB_0

About

lab_00-melh23 created by GitHub Classroom

Readme

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Languages

CMake 47.1%
C++ 40.7%
Batchfile 9.8%
C 2.4%

Search or jump to... Pulls Issues Marketplace Explore

[Unwatch](#) 2 ⚡ Star 0 ⌂ Fork 0

Code Issues Pull requests Actions Projects Security Insights

master Go to file Add file Code

melh23 Added name to README.md ... 4 minutes ago 5

- .github GitHub Classroom Autograding Workflow 21 hours ago
- _tests/_test_files Initial commit 21 hours ago
- build added .bat files 8 minutes ago
- cmake Initial commit 21 hours ago
- includes/stub Initial commit 21 hours ago
- .gitignore Initial commit 21 hours ago
- CMakeLists.txt Initial commit 21 hours ago
- README.md Added name to README.md 4 minutes ago
- main.cpp Initial commit 21 hours ago

README.md

00_LAB_0

Sassan Barkeshli

131 [100%] Linking CXX static library ../../../../lib/libgtest_main.a
 132 [100%] Built target gtest_main
 133
 134
 135 -----running basic_test.cpp-----
 136
 137
 138 [=====] Running 1 test from 1 test case.
 139 [-----] Global test environment set-up.
 140 [-----] 1 test from BASIC_TEST
 141 [RUN] BASIC_TEST.BasicTest
 142
 143
 144 after init: [-1|-1|-1|-1|-1|]
 145 [-1|-1|-1|-1|50|]
 146 [-1|-1|-1|-1|-1|50|60|]
 147 [-1|-1|-1|-1|-1|50|60|70|]
 148 [-1|-1|-1|-1|-1|50|60|70|80|]

```

149 [-1|-1|-1|-1|-1|50|60|70|80|90|]
150 found 70 at: 70
151 changing 70 to 700:
152 [-1|-1|-1|-1|-1|50|60|700|80|90|]
153
154
155 [      OK ] BASIC_TEST.BasicTest (0 ms)
156 [-----] 1 test from BASIC_TEST (0 ms total)
157
158 [-----] Global test environment tear-down
159 [=====] 1 test from 1 test case ran. (0 ms total)
160 [ PASSED ] 1 test.

161
162
163 -----running testB.cpp-----
164
165
166 [=====] Running 4 tests from 2 test cases.
167 [-----] Global test environment set-up.
168 [-----] 1 test from TEST_STUB
169 [ RUN     ] TEST_STUB.TestStub
170 [      OK ] TEST_STUB.TestStub (0 ms)
171 [-----] 1 test from TEST_STUB (0 ms total)
172
173 [-----] 3 tests from TEST_ARRAY
174 [ RUN     ] TEST_ARRAY.TestInit
175 [      OK ] TEST_ARRAY.TestInit (0 ms)
176 [ RUN     ] TEST_ARRAY.TestAppend
177 [      OK ] TEST_ARRAY.TestAppend (0 ms)
178 [ RUN     ] TEST_ARRAY.TestAt
179 [      OK ] TEST_ARRAY.TestAt (0 ms)
180 [-----] 3 tests from TEST_ARRAY (0 ms total)
181
182 [-----] Global test environment tear-down
183 [=====] 4 tests from 2 test cases ran. (1 ms total)
184 [ PASSED ] 4 tests.

185
186 ✓ testB
187

```

The screenshot shows a GitHub repository page for 'CS3A-classroom/lab_00-melh23'. The repository is private and was generated from 'CS3A-classroom/template_00_lab_0'. The page includes navigation links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The 'Actions' tab is selected, showing a green circle with a checkmark indicating a successful run. The log message says 'PASSED: testB passed all tests.' Below this, there are sections for 'GitHub Classroom Workflow' (on: push) and 'Autograding' (which is currently active). The Autograding section shows two steps: 'Set up job' (3s ago) and 'Run actions/checkout@v2' (1s ago). A 'Re-run jobs' button is also present.

unpush

30s

```
1 ► Run education/autograding@v1
2
3
4
5 ::stop-commands::1e41c4b3-f766-4da4-999e-293106bf9ee5
6
7 testB
8
9
10 -- The CXX compiler identification is GNU 7.5.0
11 -- Detecting CXX compiler ABI info
12 -- Detecting CXX compiler ABI info - done
13 -- Check for working CXX compiler: /usr/bin/c++ - skipped
14 -- Detecting CXX compile features
15 -- Detecting CXX compile features - done
16 -- Configuring done
17 -- Generating done
18 -- Build files have been written to: /home/runner/work/lab_00-melh23/lab_00-
melh23/googletest
19 Scanning dependencies of target googletest
20 [ 11%] Creating directories for 'googletest'
21 [ 22%] Performing download step (git clone) for 'googletest'
22 Cloning into 'googletest-src'...
23 Note: switching to 'release-1.8.0'.
24
25 You are in 'detached HEAD' state. You can look around, make experimental
26 changes and commit them, and you can discard any commits you make in this
27 state without impacting any branches by switching back to a branch.
28
29 If you want to create a new branch to retain commits you create, you may
30 do so (now or later) by using -c with the switch command. Example:
31
32 git switch -c <new-branch-name>
33
34 Or undo this operation with:
35
36 git switch -
37
38 Turn off this advice by setting config variable advice.detachedHead to false
39
40 HEAD is now at ec44c6c1 Merge pull request #821 from mazong1123/master
41 [ 33%] Performing update step for 'googletest'
42 [ 44%] No patch step for 'googletest'
43 [ 55%] No configure step for 'googletest'
44 [ 66%] No build step for 'googletest'
45 [ 77%] No install step for 'googletest'
46 [ 88%] No test step for 'googletest'
47 [100%] Completed 'googletest'           ...
48 [100%] Built target googletest
49 CMake Deprecation Warning at googletest/googletest-src/CMakeLists.txt:1
(cmake_minimum_required):
50   Compatibility with CMake < 2.8.12 will be removed from a future version of
51   CMake.
52
53   Update the VERSION argument <min> value or use a ...<max> suffix to tell
54   CMake that the project does not need compatibility with older versions.
55
56
57 -- The C compiler identification is GNU 7.5.0
58 -- Detecting C compiler ABI info
59 -- Detecting C compiler ABI info - done
60 -- Check for working C compiler: /usr/bin/cc - skipped
61 -- Detecting C compile features
62 -- Detecting C compile features - done
63 CMake Deprecation Warning at googletest/googletest-src/gtest/CMakeLists.txt:41
(cmake_minimum_required):
64   Compatibility with CMake < 2.8.12 will be removed from a future version of
65   CMake.
66
67   Update the VERSION argument <min> value or use a ...<max> suffix to tell
68   CMake that the project does not need compatibility with older versions.
69
70
71 CMake Deprecation Warning at googletest/googletest-src/gtest/CMakeLists.txt:48
(cmake_minimum_required):
72   Compatibility with CMake < 2.8.12 will be removed from a future version of
73   CMake.
74
75   Update the VERSION argument <min> value or use a ...<max> suffix to tell
76   CMake that the project does not need compatibility with older versions.
77
78
```

```

79  -- Found PythonInterp: /usr/bin/python (found version "2.7.17")
80  -- Looking for pthread.h
81  -- Looking for pthread.h - found
82  -- Performing Test CMAKE_HAVE_LIBC_PTHREAD
83  -- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Failed
84  -- Looking for pthread_create in pthreads
85  -- Looking for pthread_create in pthreads - not found
86  -- Looking for pthread_create in pthread
87  -- Looking for pthread_create in pthread - found
88  -- Found Threads: TRUE
89  -- Configuring done
90  -- Generating done
91  -- Build files have been written to: /home/runner/work/lab_00-melh23/lab_00-melh23
92
93  Scanning dependencies of target main
94  [ 4%] Building CXX object CMakeFiles/main.dir/main.cpp.o
95  [ 8%] Building CXX object CMakeFiles/main.dir/includes/stub/stub.cpp.o
96  [ 12%] Linking CXX executable bin/main
97  [ 12%] Built target main
98  Scanning dependencies of target gtest
99  [ 16%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/gtest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
100 [ 20%] Linking CXX static library ../../lib/libgtest.a
101 [ 20%] Built target gtest
102  Scanning dependencies of target basic_test
103 [ 25%] Building CXX object
CMakeFiles/basic_test.dir/_tests/_test_files/basic_test.cpp.o
104 [ 29%] Building CXX object
CMakeFiles/basic_test.dir/includes/array_functions/array_functions.cpp.o
105 [ 33%] Linking CXX executable bin/basic_test
106 [ 33%] Built target basic_test
107  Scanning dependencies of target testA
108 [ 37%] Building CXX object CMakeFiles/testA.dir/_tests/_test_files/testA.cpp.o
109 [ 41%] Building CXX object
CMakeFiles/testA.dir/includes/array_functions/array_functions.cpp.o
110 [ 45%] Linking CXX executable bin/testA
111 [ 45%] Built target testA
112  Scanning dependencies of target testB
113 [ 50%] Building CXX object CMakeFiles/testB.dir/_tests/_test_files/testB.cpp.o
114 [ 54%] Building CXX object
CMakeFiles/testB.dir/includes/array_functions/array_functions.cpp.o
115 [ 58%] Building CXX object CMakeFiles/testB.dir/includes/stub/stub.cpp.o
116 [ 62%] Linking CXX executable bin/testB
117 [ 62%] Built target testB
118  Scanning dependencies of target gmock_main
119 [ 66%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/CMakeFiles/gmock_main.dir/_/gtest-all.cc.o
120 [ 70%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/CMakeFiles/gmock_main.dir/src/gmock-all.cc.o
121 [ 75%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/CMakeFiles/gmock_main.dir/src/gmock_main.cc.o
122 [ 79%] Linking CXX static library ../../lib/libgmock_main.a
123 [ 79%] Built target gmock_main
124  Scanning dependencies of target gmock
125 [ 83%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/CMakeFiles/gmock.dir/_/gtest-all.cc.o
126 [ 87%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/CMakeFiles/gmock.dir/src/gmock-all.cc.o
127 [ 91%] Linking CXX static library ../../lib/libgmock.a
128 [ 91%] Built target gmock
129  Scanning dependencies of target gtest_main
130 [ 95%] Building CXX object googlemock/gtest/googlemock-
build/googlemock/gtest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.o
131 [100%] Linking CXX static library ../../lib/libgtest_main.a
132 [100%] Built target gtest_main
133
134  -----running basic_test.cpp-----
135
136
137
138 [=====] Running 1 test from 1 test case.
139 [-----] Global test environment set-up.
140 [-----] 1 test from BASIC_TEST
141 [ RUN ] BASIC_TEST.BasicTest
142
143
144 after init: [-1|-1|-1|-1|-1|]
145 [-1|-1|-1|-1|50|]
146 [-1|-1|-1|-1|50|60|]
147 [-1|-1|-1|-1|50|60|70|]
148 [-1|-1|-1|-1|50|60|70|80|]
149 [-1|-1|-1|-1|50|60|70|80|90|]

```

```
150  found 70 at: 70
151  changing 70 to 700:
152  [-1|-1|-1|-1|50|60|700|80|90|]
153
154
155  [      OK ] BASIC_TEST.BasicTest (0 ms)
156  [-----] 1 test from BASIC_TEST (0 ms total)
157
158  [-----] Global test environment tear-down
159  [=====] 1 test from 1 test case ran. (0 ms total)
160  [ PASSED ] 1 test.

161
162
163  -----running testB.cpp-----
164
165
166  [=====] Running 4 tests from 2 test cases.
167  [-----] Global test environment set-up.
168  [-----] 1 test from TEST_STUB
169  [ RUN    ] TEST_STUB.TestStub
170  [      OK ] TEST_STUB.TestStub (0 ms)
171  [-----] 1 test from TEST_STUB (0 ms total)

172
173  [-----] 3 tests from TEST_ARRAY
174  [ RUN    ] TEST_ARRAY.TestInit
175  [      OK ] TEST_ARRAY.TestInit (0 ms)
176  [ RUN    ] TEST_ARRAY.TestAppend
177  [      OK ] TEST_ARRAY.TestAppend (0 ms)
178  [ RUN    ] TEST_ARRAY.TestAt
179  [      OK ] TEST_ARRAY.TestAt (0 ms)
180  [-----] 3 tests from TEST_ARRAY (0 ms total)

181
182  [-----] Global test environment tear-down
183  [=====] 4 tests from 2 test cases ran. (1 ms total)
184  [ PASSED ] 4 tests.

185
186  ✅ testB
187
```

The screenshot shows a GitHub repository page for 'CS3A-classroom / lab_00-melh23'. The repository is private and was generated from 'CS3A-classroom/template_00_lab_0'. The main navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the right, there are buttons for 'Unwatch', 'Star', 'Fork', and a dropdown menu.

The repository has 2 pull requests, 0 stars, and 0 forks. The 'Actions' tab is selected, showing a green checkmark indicating 'PASSED: testB passed all tests.' Below this, the 'GitHub Classroom Workflow' is shown, triggered by a 'push' event. The 'Autograding' section is expanded, showing the log output for the 'Run education/autograding@v1' job. The log details the execution of CMake and the running of testB.cpp, which passed all 4 tests. The log concludes with 'All tests passed' and a series of emojis. Two additional actions are listed at the bottom: 'Post Run actions/checkout@v2' and 'Complete job'.

```
136
137
138 [=====] Running 1 test from 1 test case.
139 [-----] Global test environment set-up.
140 [-----] 1 test from BASIC_TEST
141 [ RUN ] BASIC_TEST.BasicTest
142
143
144 after init: [-1|-1|-1|-1|-1|]
145 [-1|-1|-1|-1|50|]
146 [-1|-1|-1|-1|50|60|]
147 [-1|-1|-1|-1|50|60|70|]
148 [-1|-1|-1|-1|50|60|70|80|]
149 [-1|-1|-1|-1|50|60|70|80|90|]
150 found 70 at: 70
151 changing 70 to 700:
152 [-1|-1|-1|-1|50|60|700|80|90|]
153
154
155 [     OK ] BASIC_TEST.BasicTest (0 ms)
156 [-----] 1 test from BASIC_TEST (0 ms total)
157
158 [-----] Global test environment tear-down
159 [=====] 1 test from 1 test case ran. (0 ms total)
160 [ PASSED ] 1 test.
161
162
163 -----running testB.cpp-----
164
165
166 [=====] Running 4 tests from 2 test cases.
167 [-----] Global test environment set-up.
168 [-----] 1 test from TEST_STUB
169 [ RUN ] TEST_STUB.TestStub
170 [     OK ] TEST_STUB.TestStub (0 ms)
171 [-----] 1 test from TEST_STUB (0 ms total)
172
173 [-----] 3 tests from TEST_ARRAY
174 [ RUN ] TEST_ARRAY.TestInit
175 [     OK ] TEST_ARRAY.TestInit (0 ms)
176 [ RUN ] TEST_ARRAY.TestAppend
177 [     OK ] TEST_ARRAY.TestAppend (0 ms)
178 [ RUN ] TEST_ARRAY.TestAt
179 [     OK ] TEST_ARRAY.TestAt (0 ms)
180 [-----] 3 tests from TEST_ARRAY (0 ms total)
181
182 [-----] Global test environment tear-down
183 [=====] 4 tests from 2 test cases ran. (1 ms total)
184 [ PASSED ] 4 tests.
185
186 [ checked ] testB
187
188
189 ::1e41c4b3-f766-4da4-999e-293106bf9ee5::
190
191 All tests passed
192
193 ✨ ★ ❤️ 💙 💙 💙 💙 💙 💙 💙 💙 💙 💙 💙 ✨
194
195 Points 50/50
196
```

