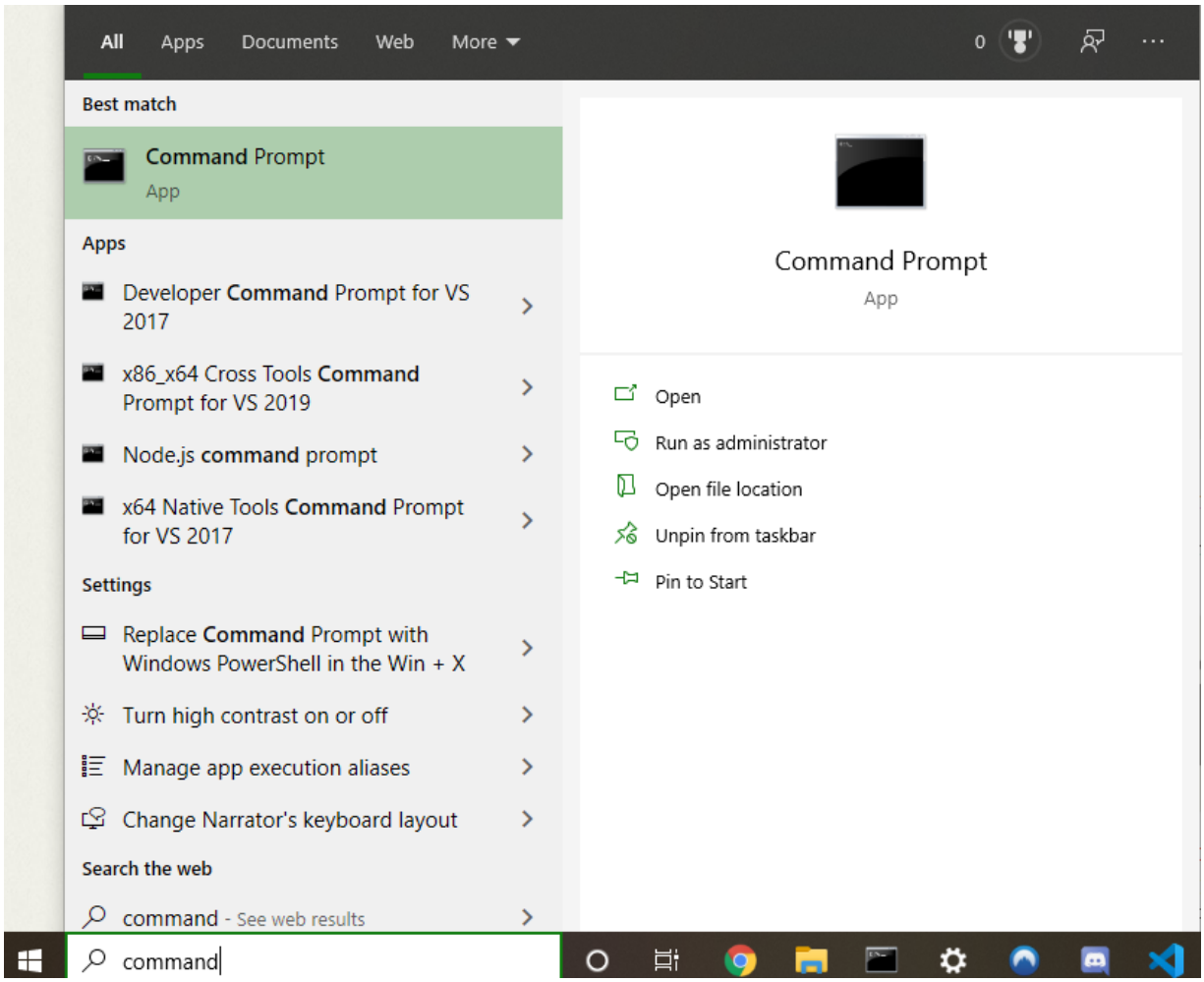
Windows Insructions

Opening the terminal

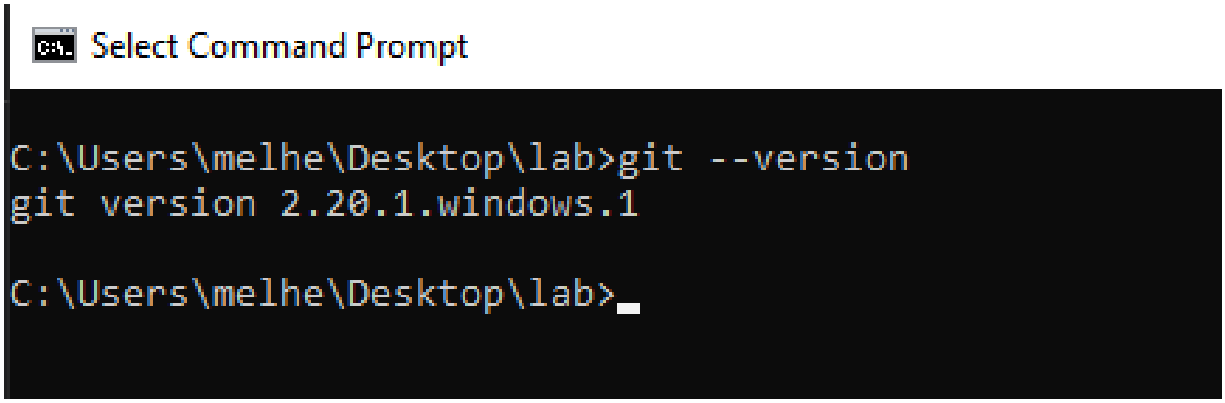


Installing git

is `git` installed?

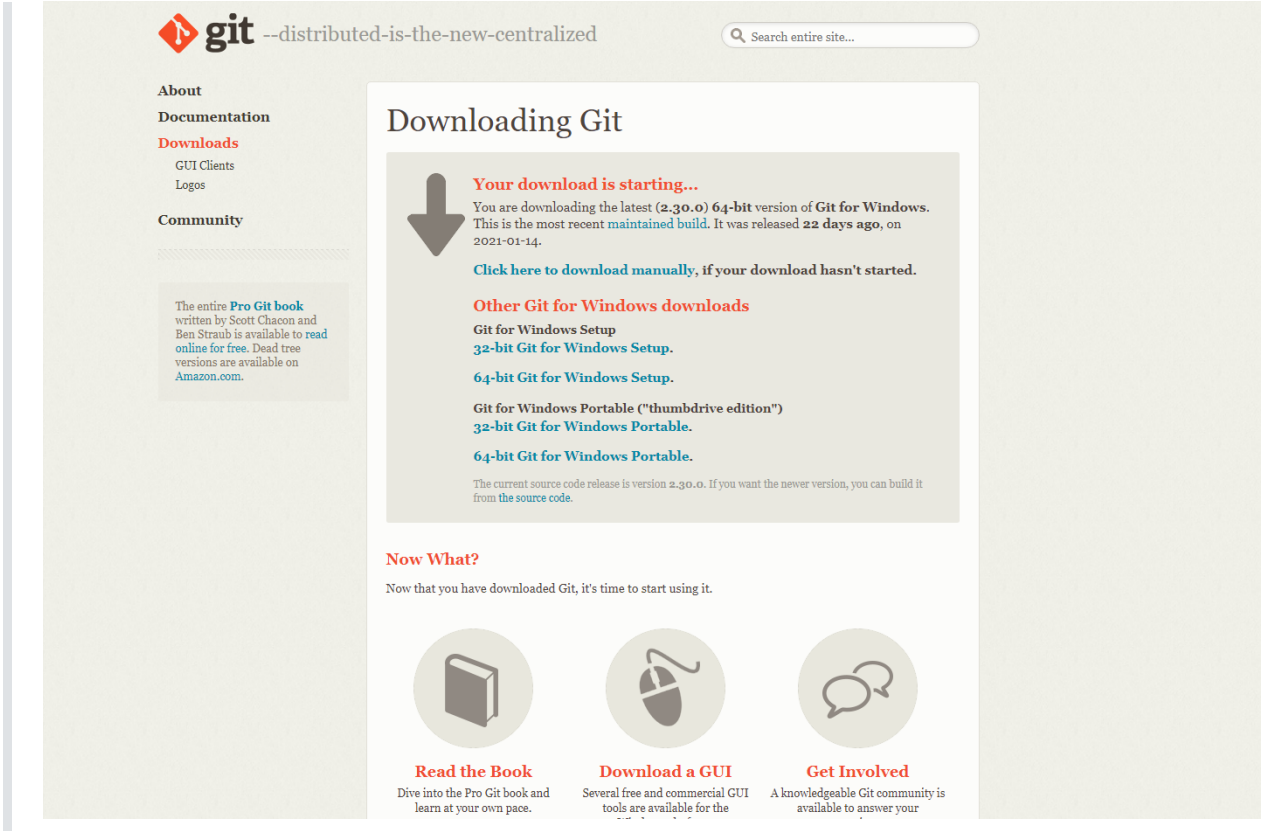
Let's check to see if `git` is installed on your system: type `git --version` at the commandline.

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.

> 
>

Download `git`

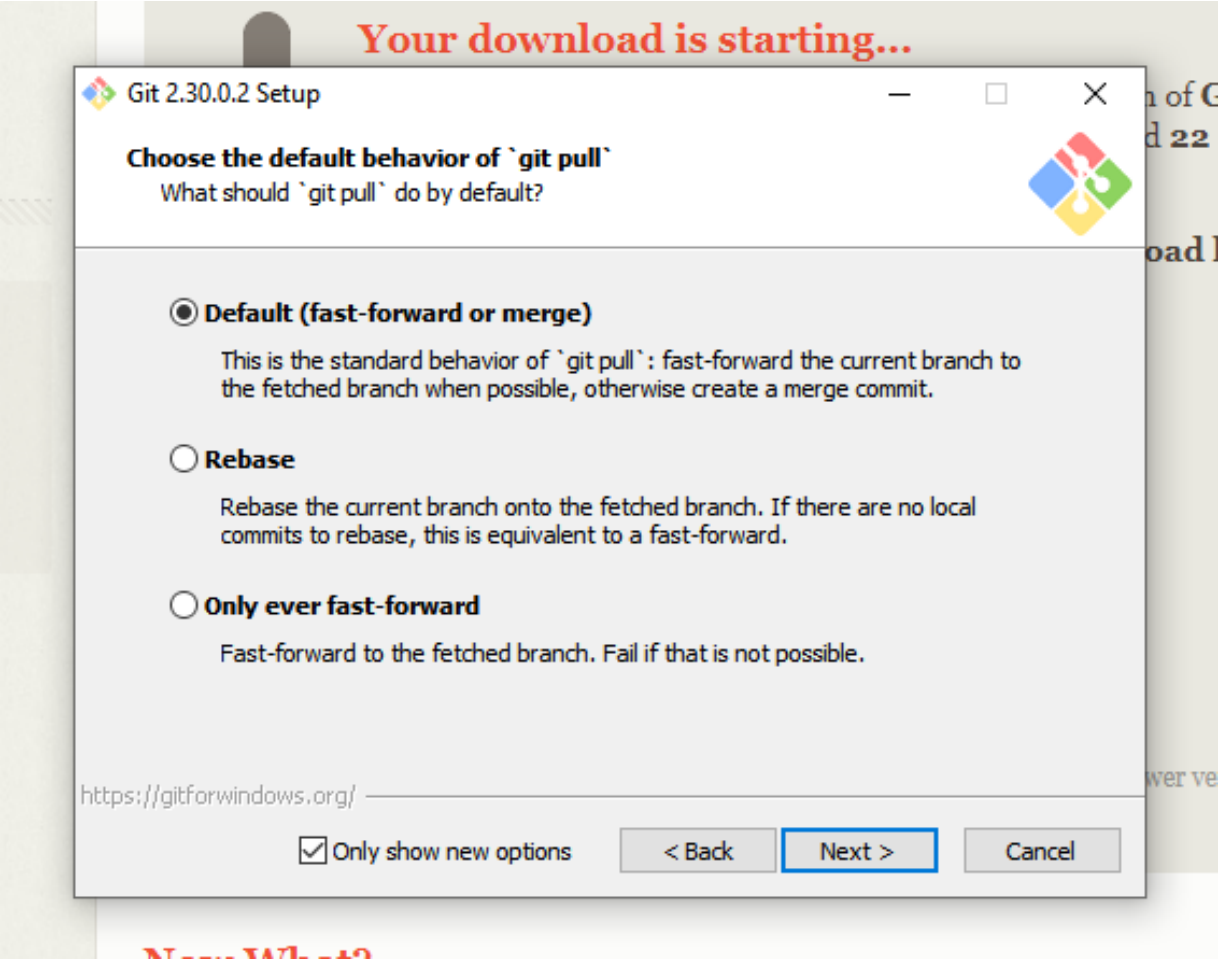
Download git from here. You should get an exe similar to this: `Git-2.30.0.2-64-bit.exe`



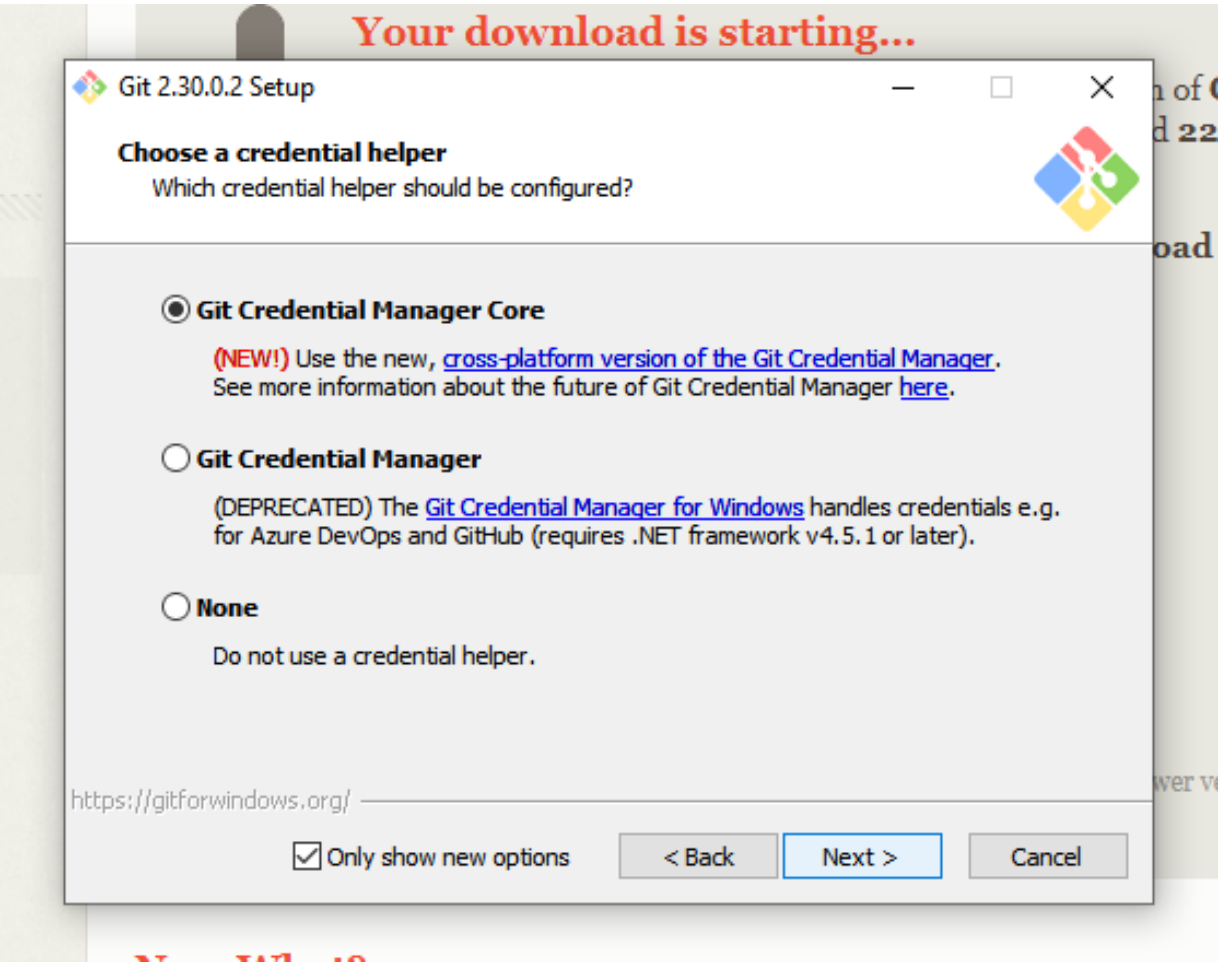
Install `git`

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

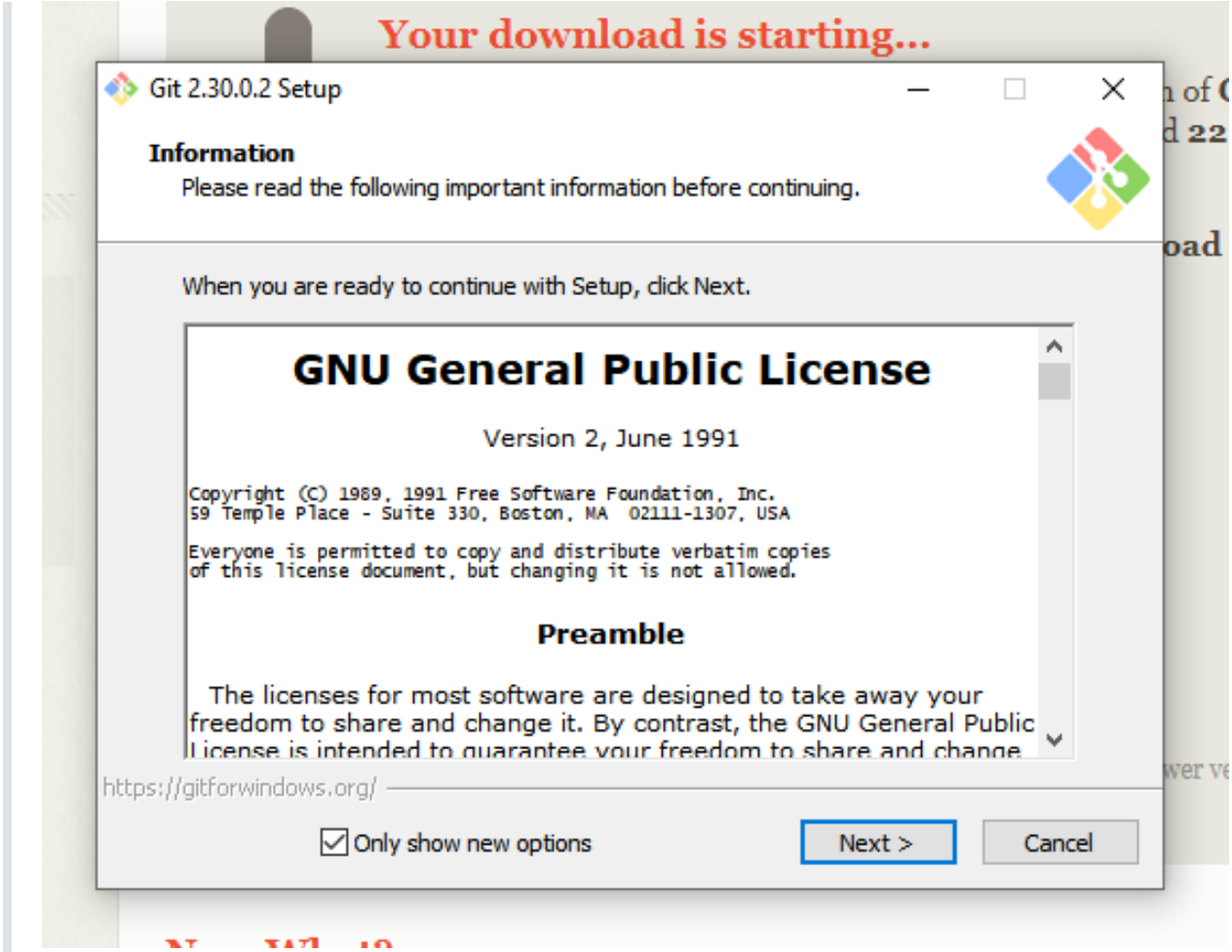
Choose Default.



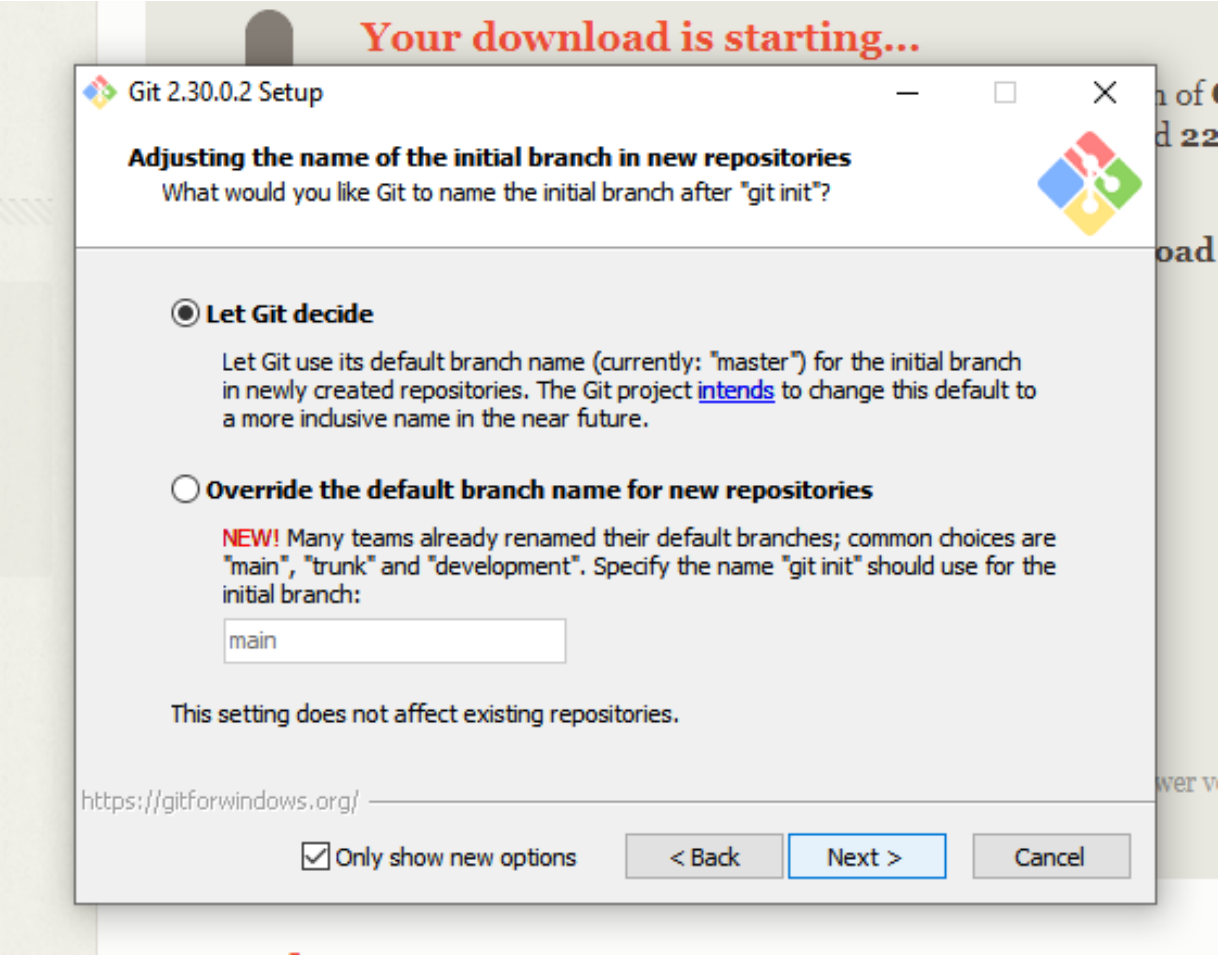
Choose Git Credential Manager Core.



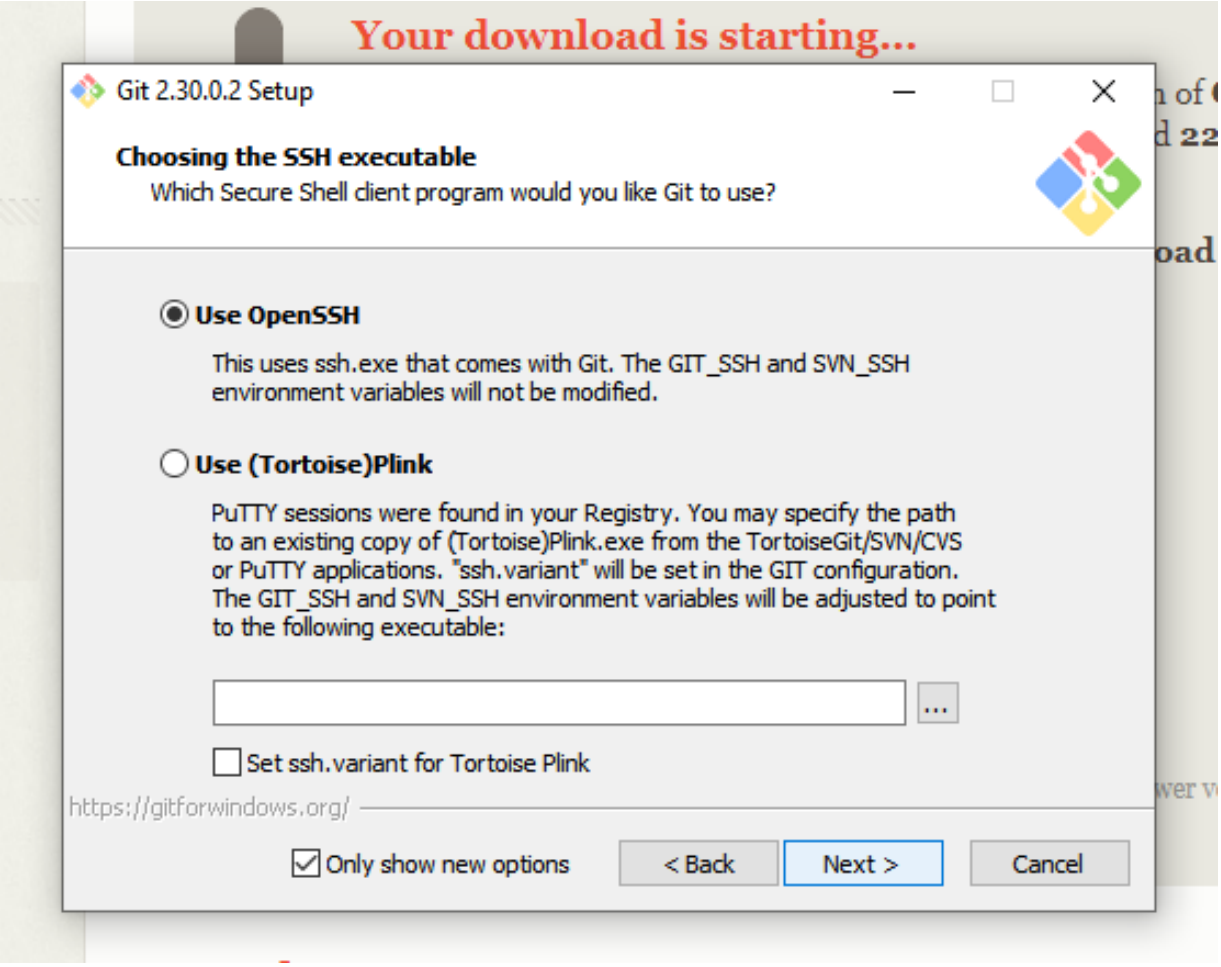
Select Only show new options if not selected, and click next.



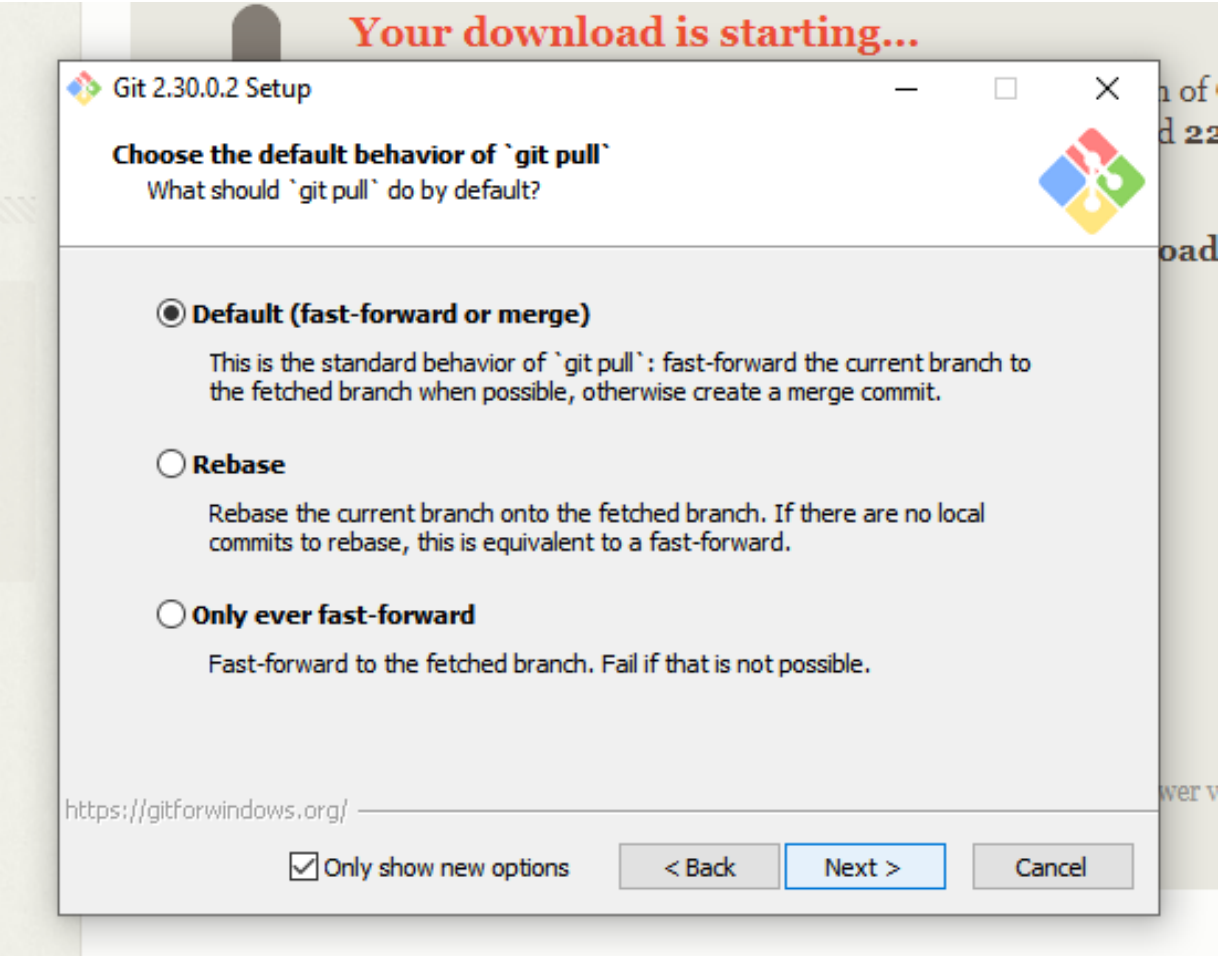
Choose Let git decide.



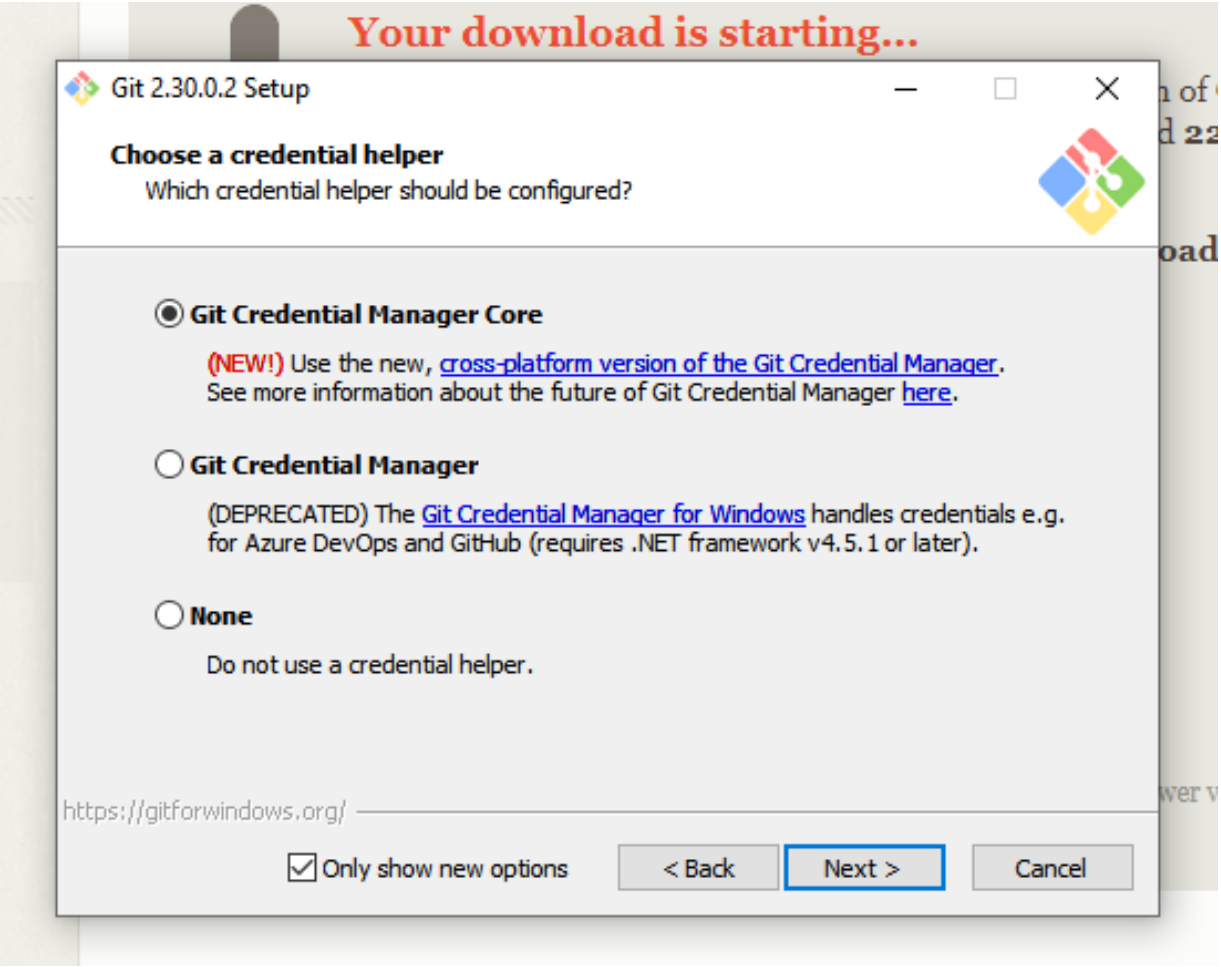
Choose Use OpenSSH.



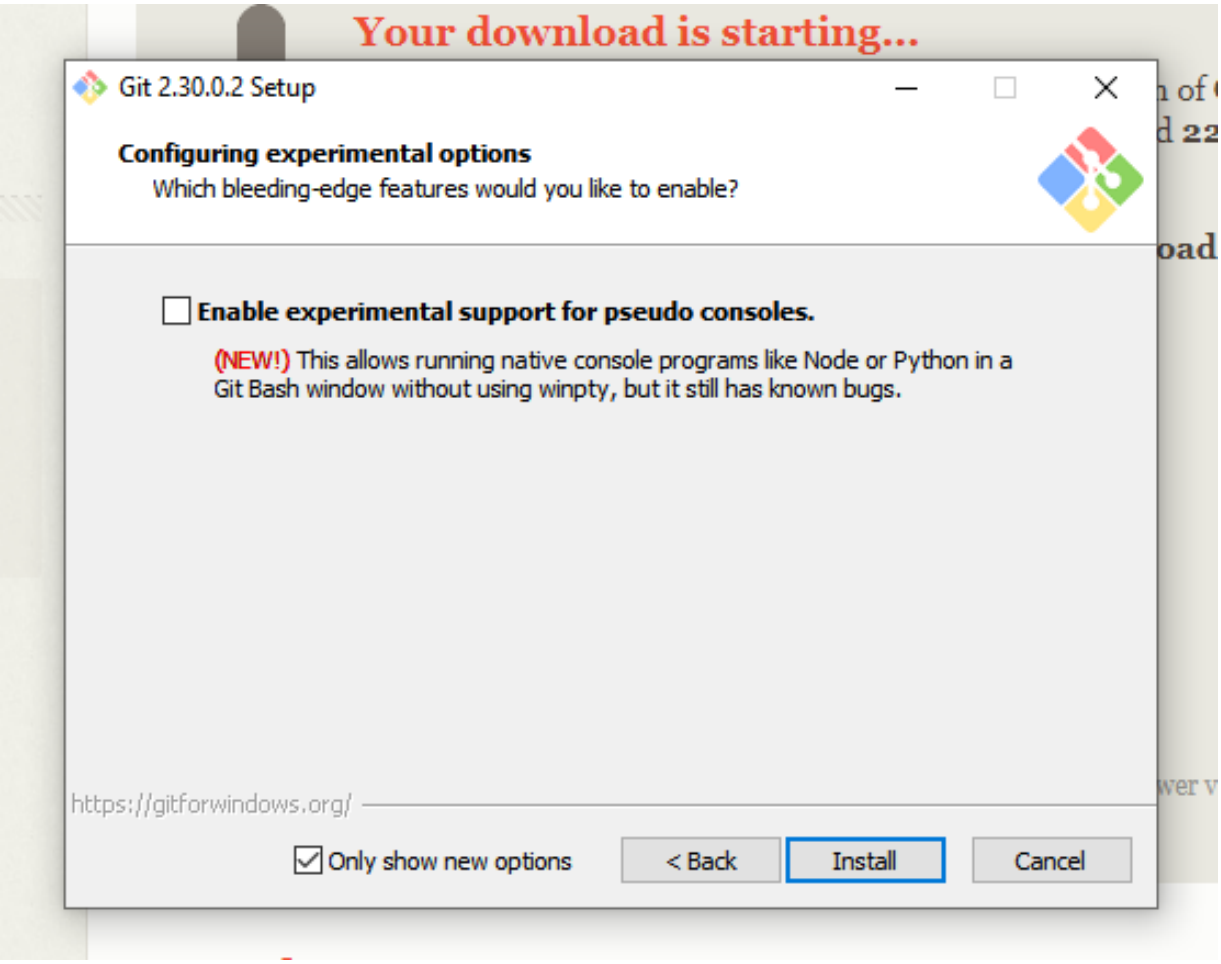
Choose Default (fast-forward or merge).

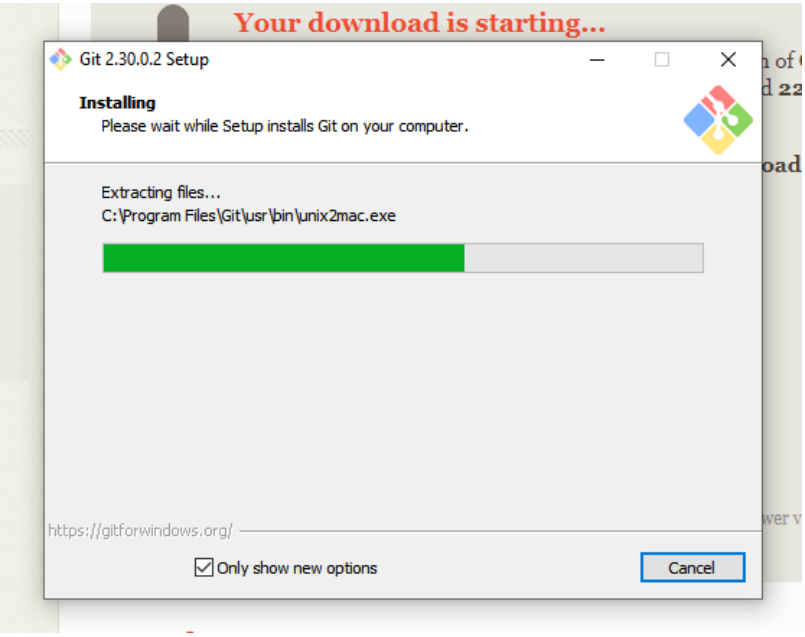


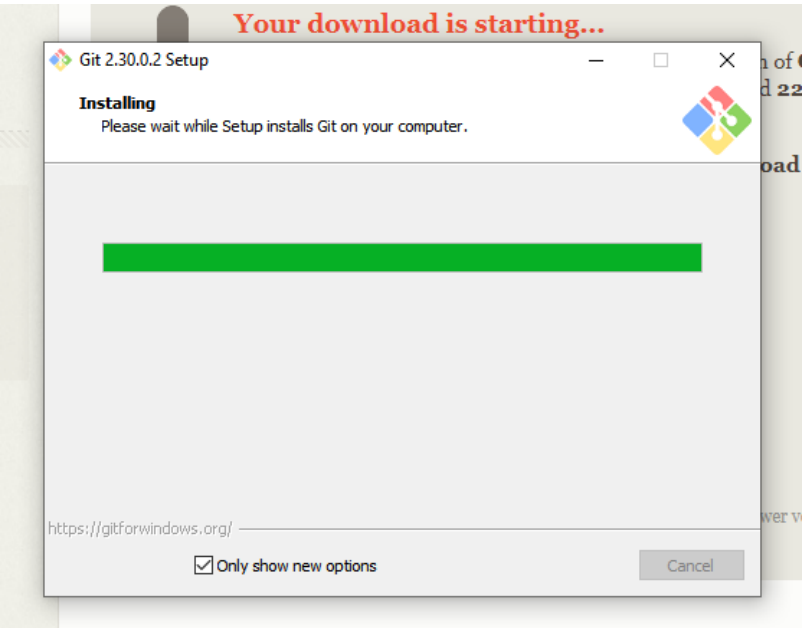
Choose Git Credential Manager Core


Your download is starting...
Git 2.30.0.2 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager Core
(NEW!) Use the new, cross-platform version of the Git Credential Manager.
See more information about the future of Git Credential Manager here.
Git Credential Manager
(DEPRECATED) The Git Credential Manager for Windows handles credentials e.g.
for Azure DevOps and GitHub (requires .NET framework v4.5.1 or later).
None
Do not use a credential helper.
https://gitforwindows.org/
Only show new options
< Back
Next >
Cancel

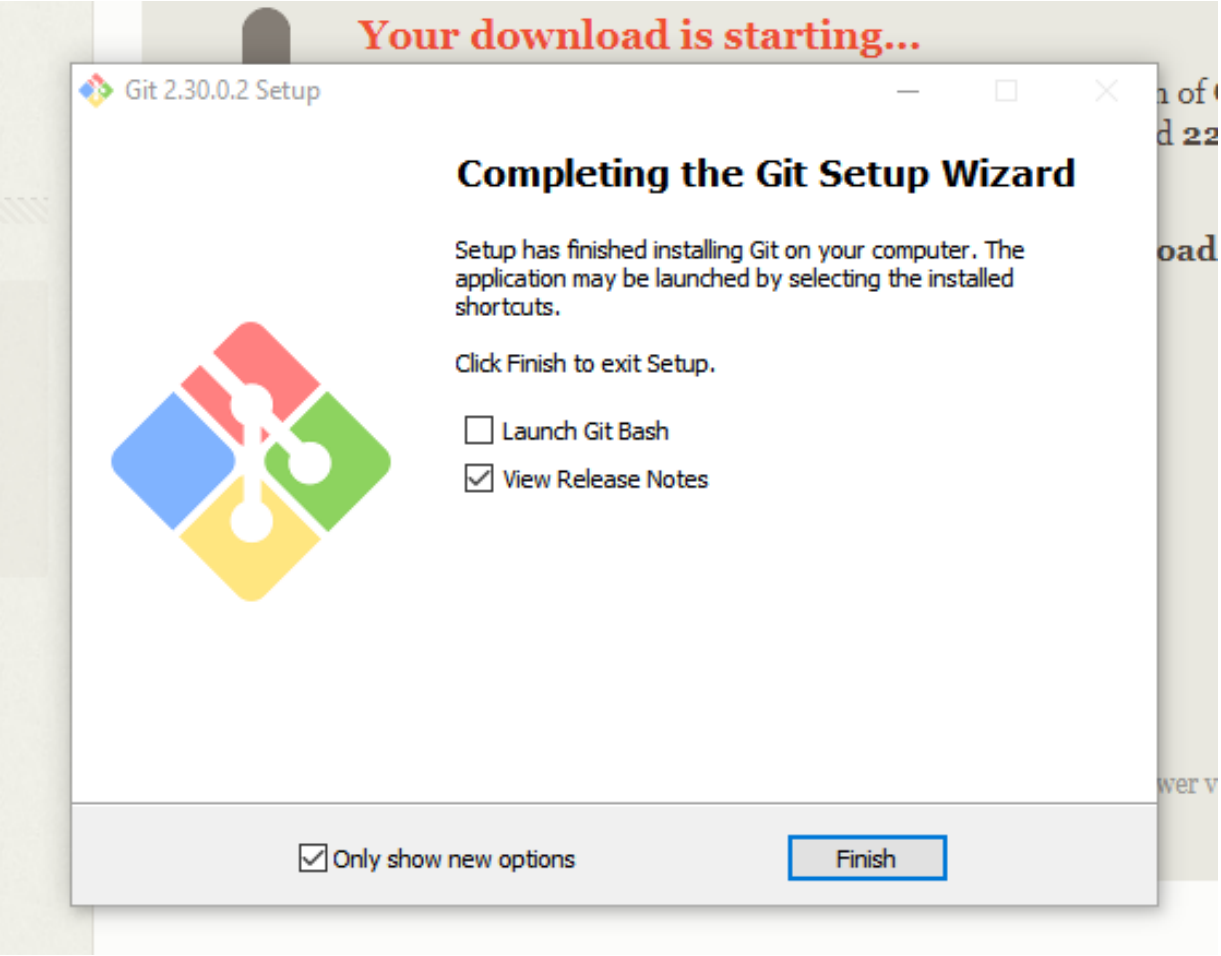
And Install!


Your download is starting...
Git 2.30.0.2 Setup
Configuring experimental options
Which bleeding-edge features would you like to enable?
Enable experimental support for pseudo consoles.
(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.
https://gitforwindows.org/
Only show new options
< Back
Install
Cancel

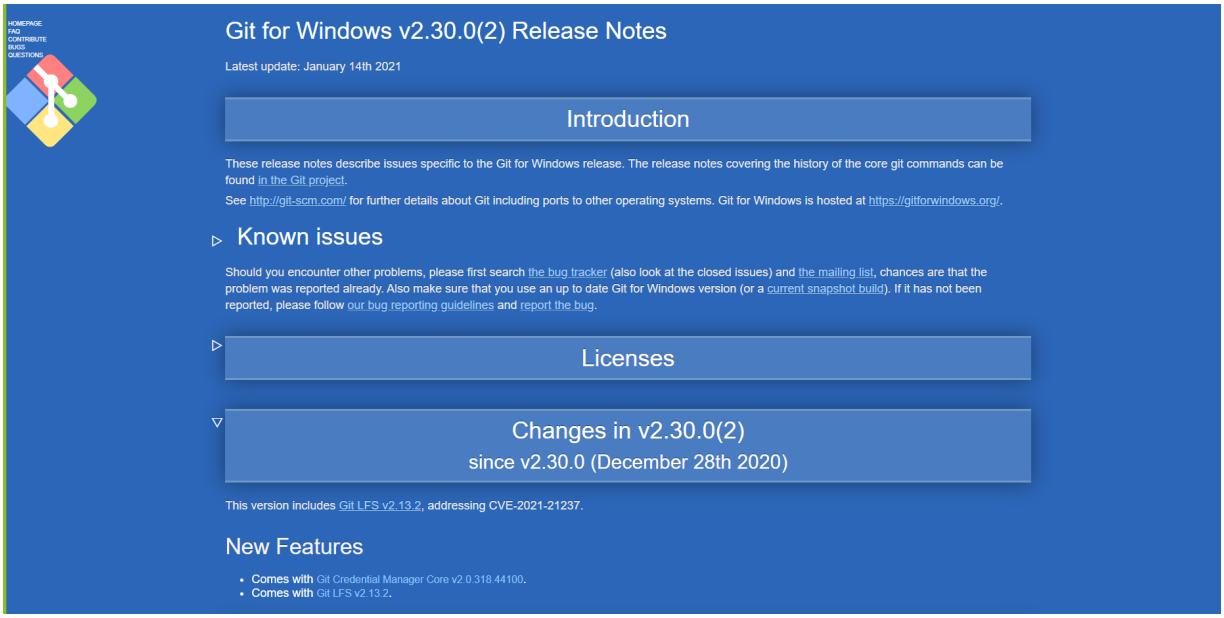




When it is done installing, click Finish.

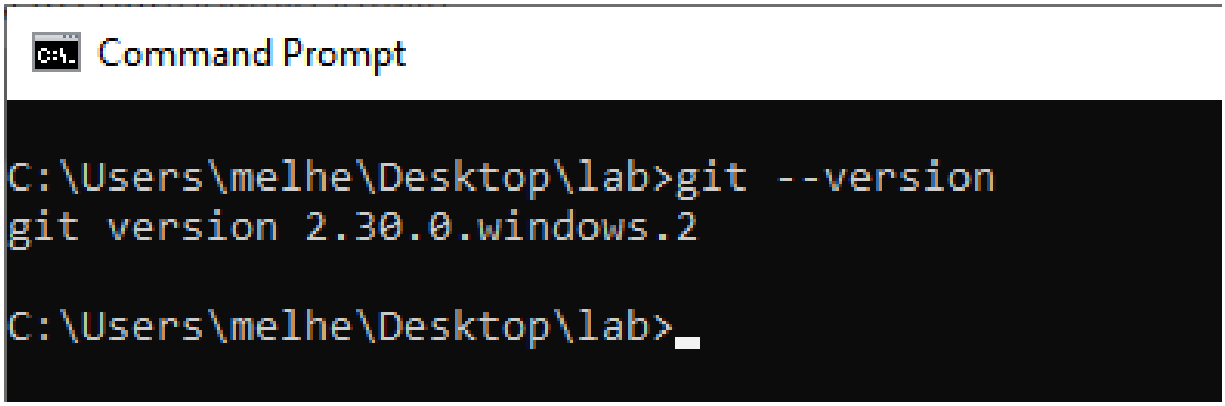


You may view release notes if you want.



Check the version of the git again:

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

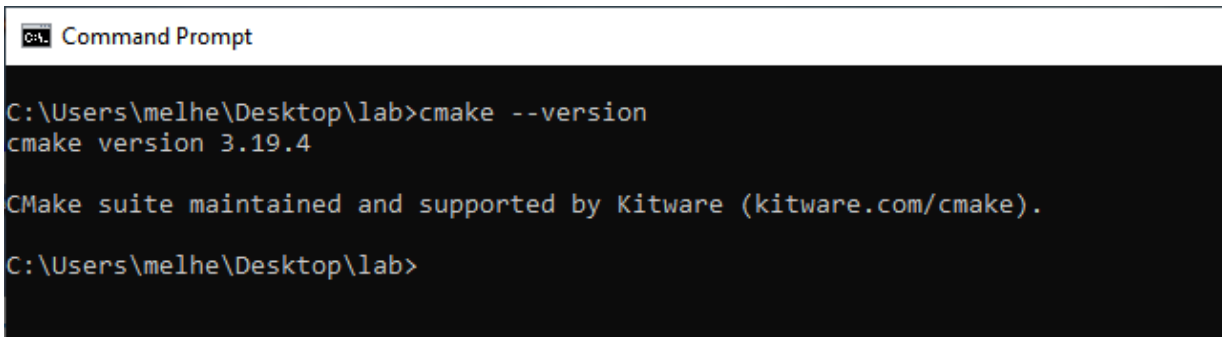


Installing cmake

is `cmake` installed?

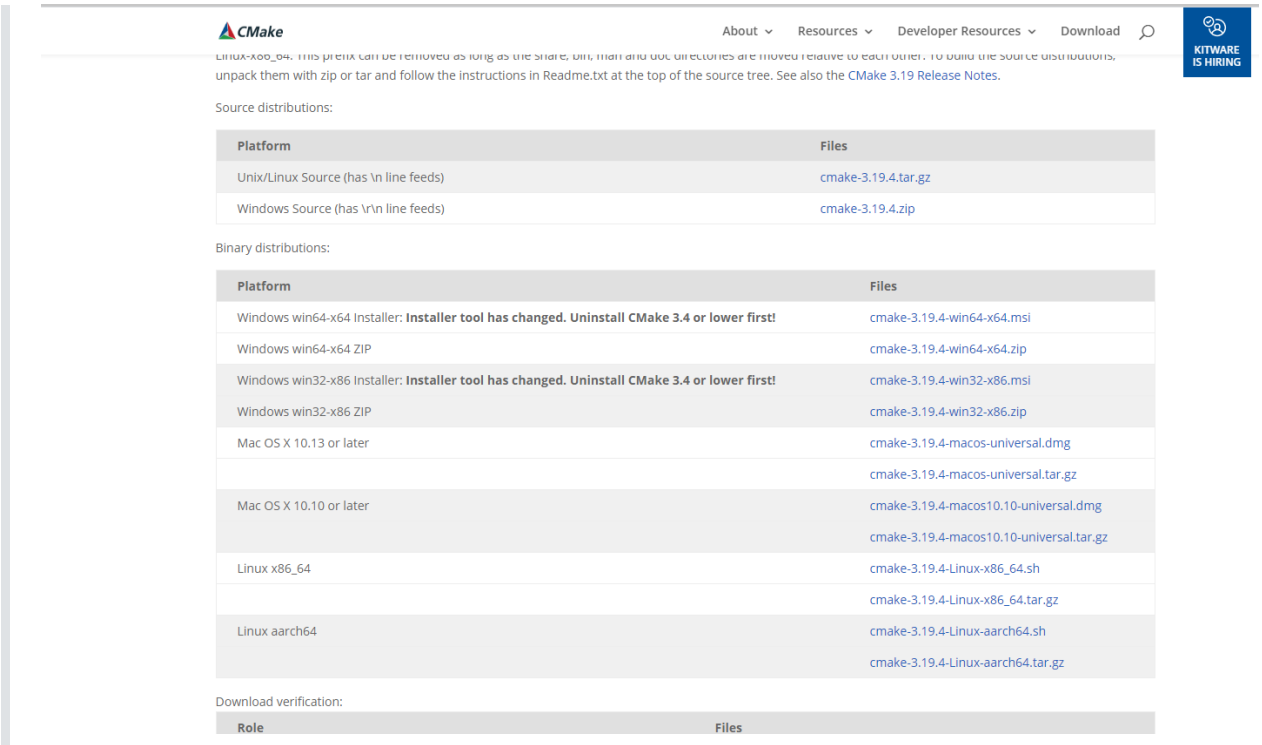
Let's check to see if `cmake` is installed on your system: type `cmake --version` at the commandline.

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.



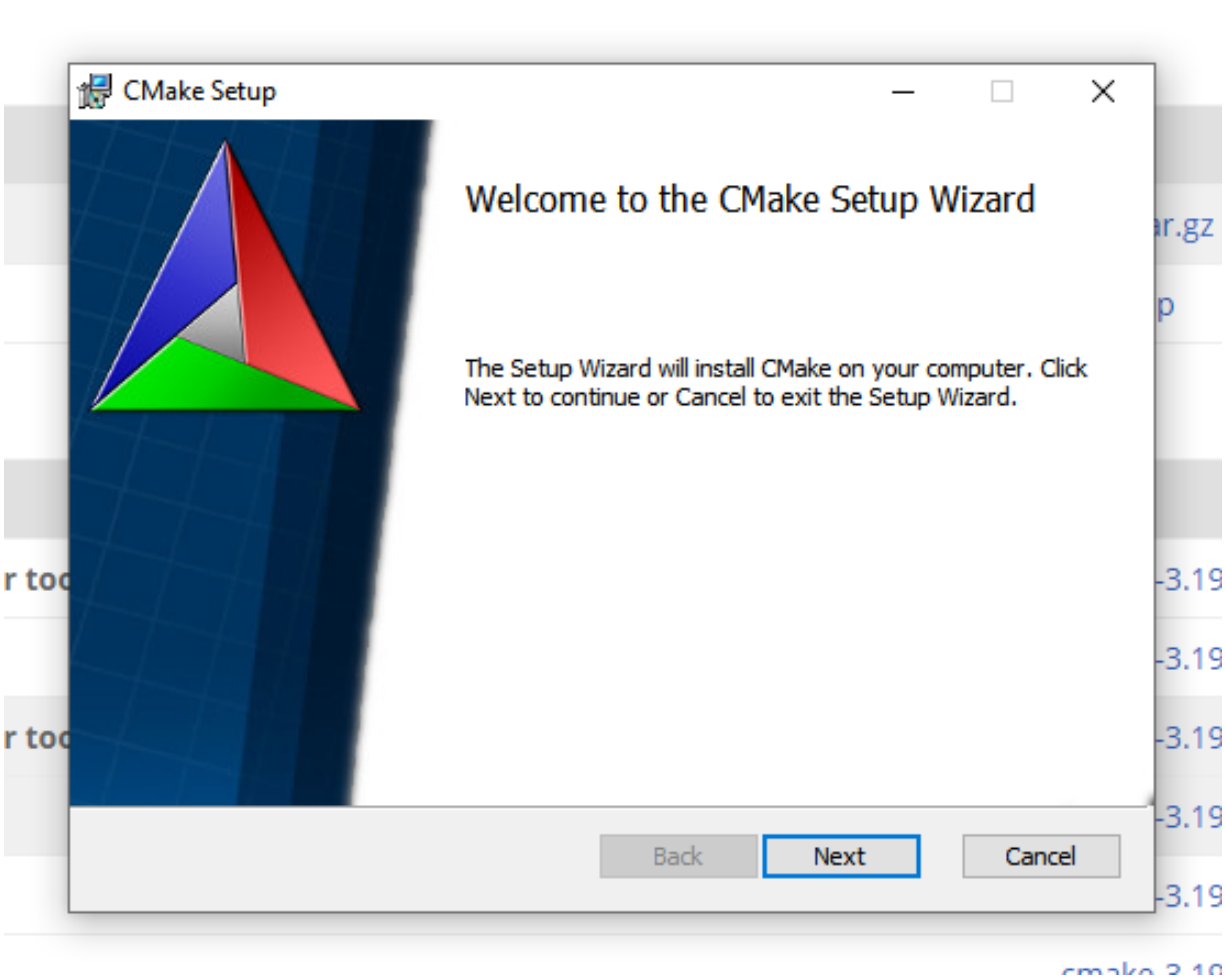
Download `cmake`

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`

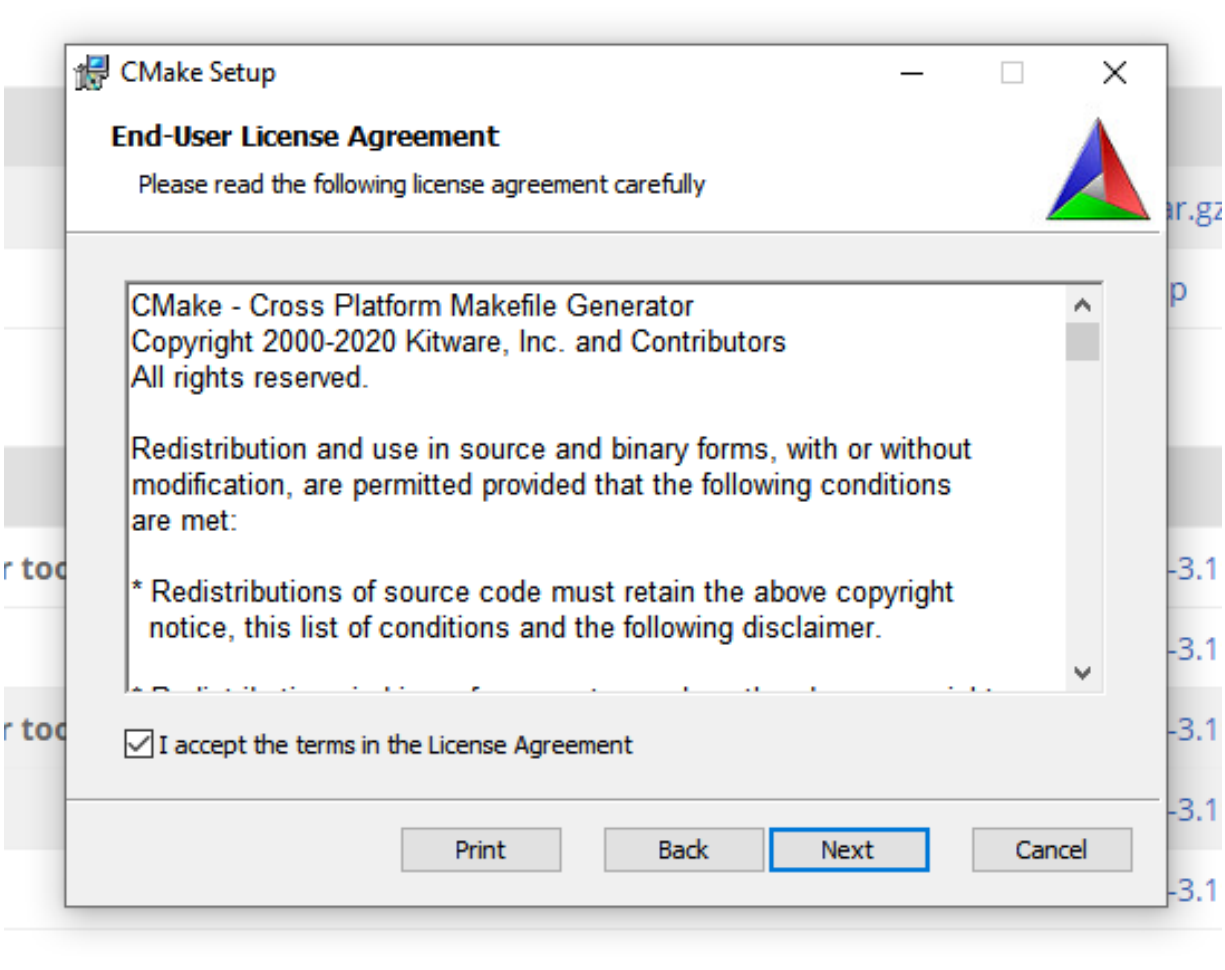


Install `cmake`

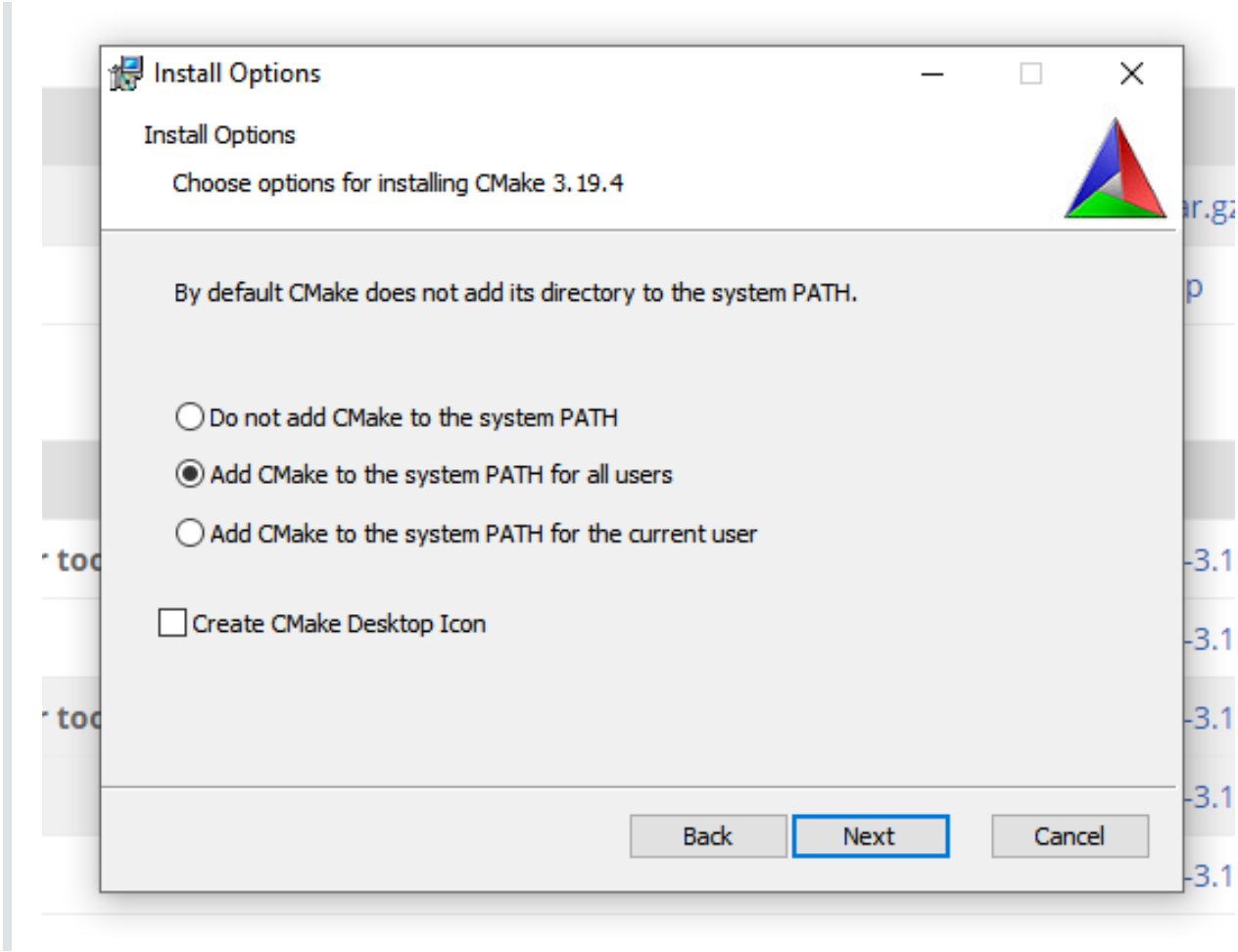
Open/run the executable, and follow the steps to install.



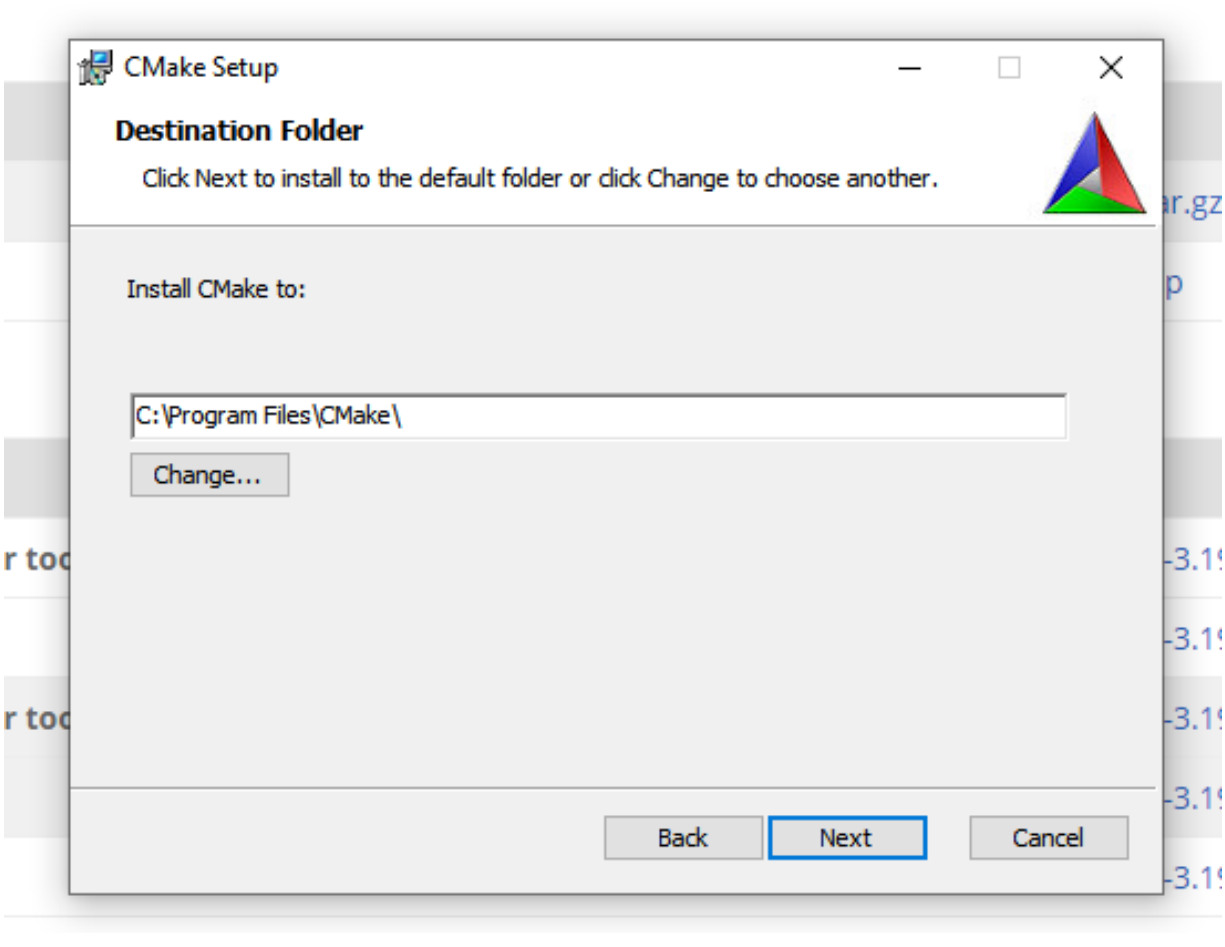
Check I accept the terms in the Liscense Agreement.



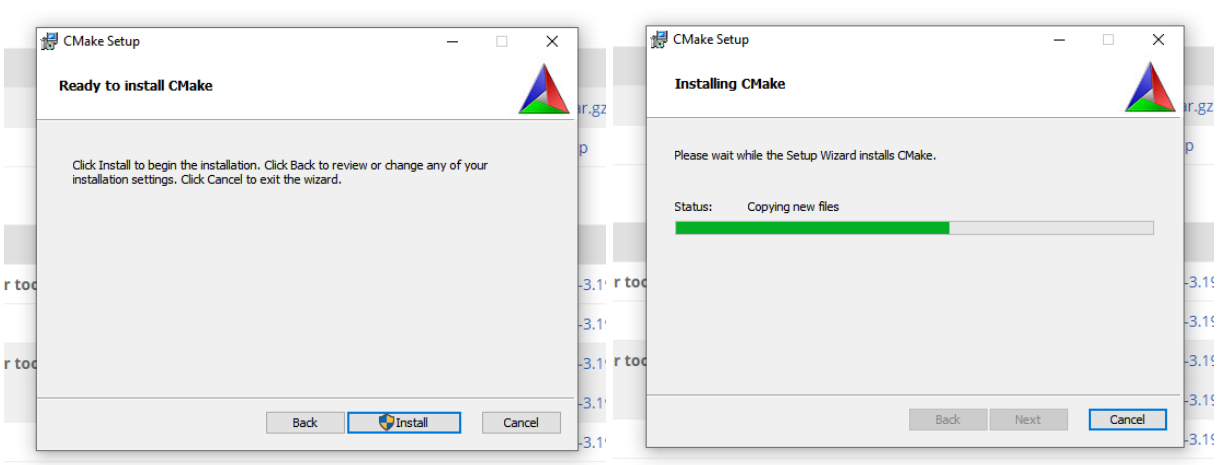
Make sure to select Add CMake to the system PATH for all users. You can create a Desktop icon if you want, but you will not need to use it for this class.



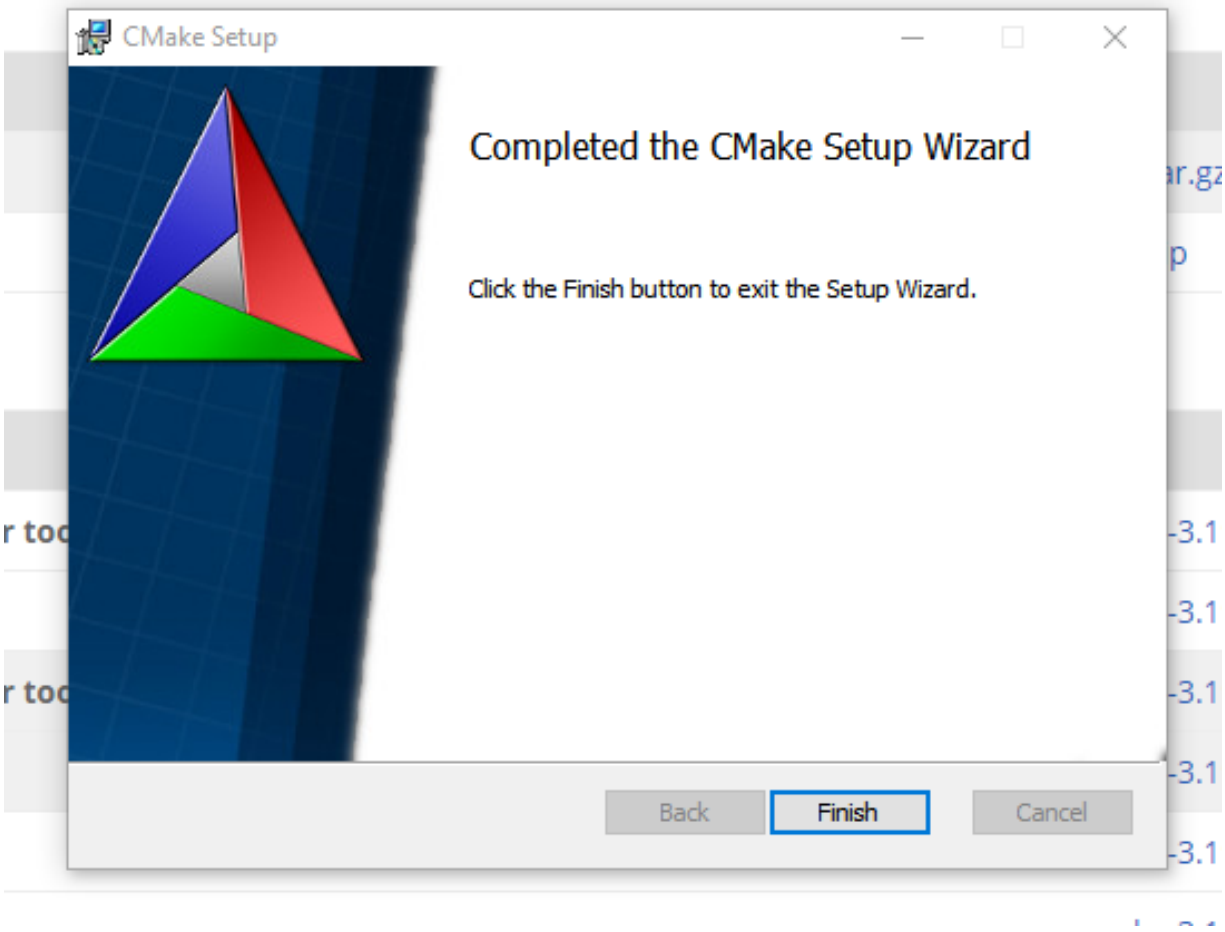
The default install location wshould be C:\Program Files\CMake



Install!

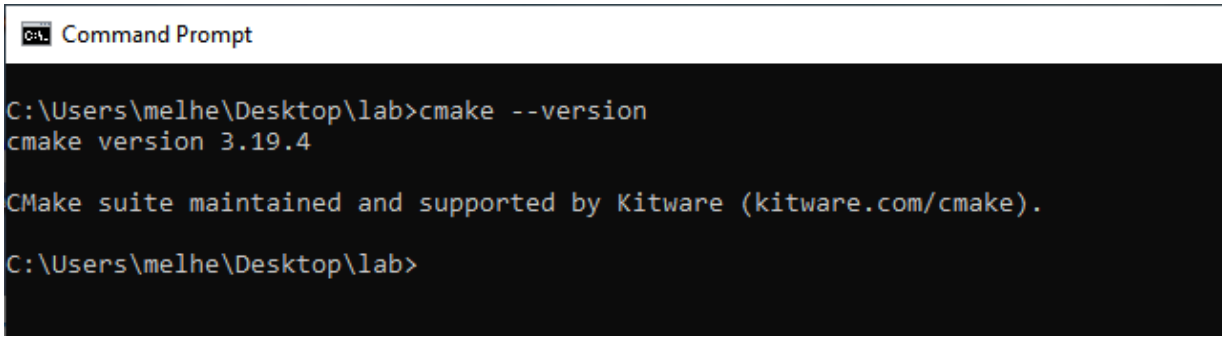


When it is done installing, click Finish.



Check the version of the git again:

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

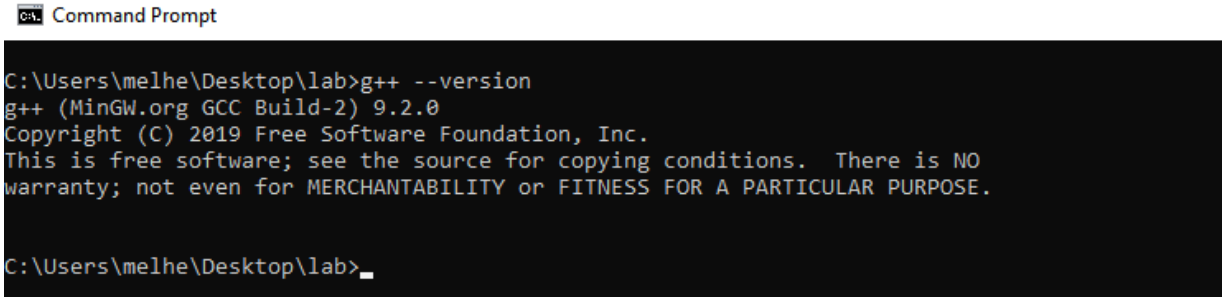


Install MinGW

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.

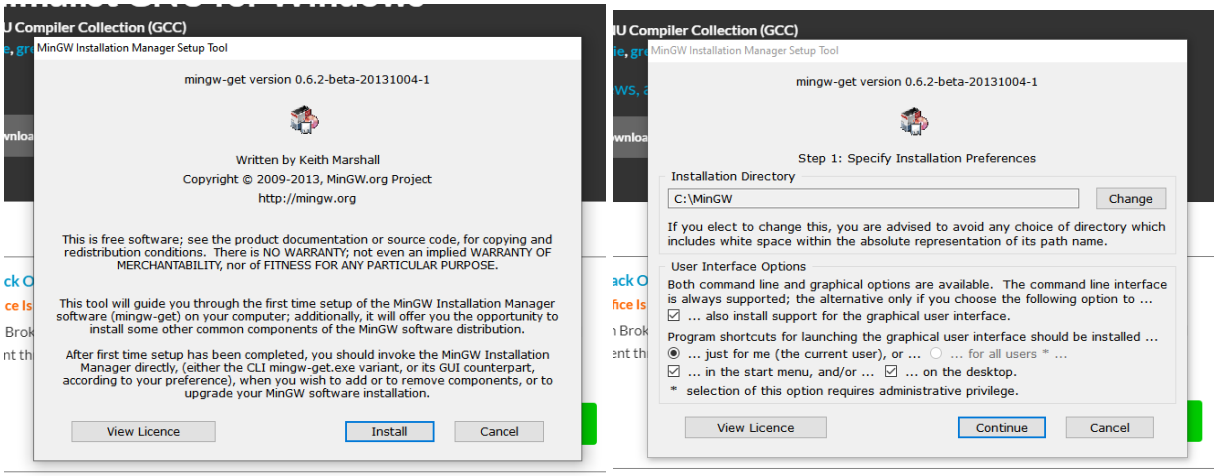


Download `MinGW`

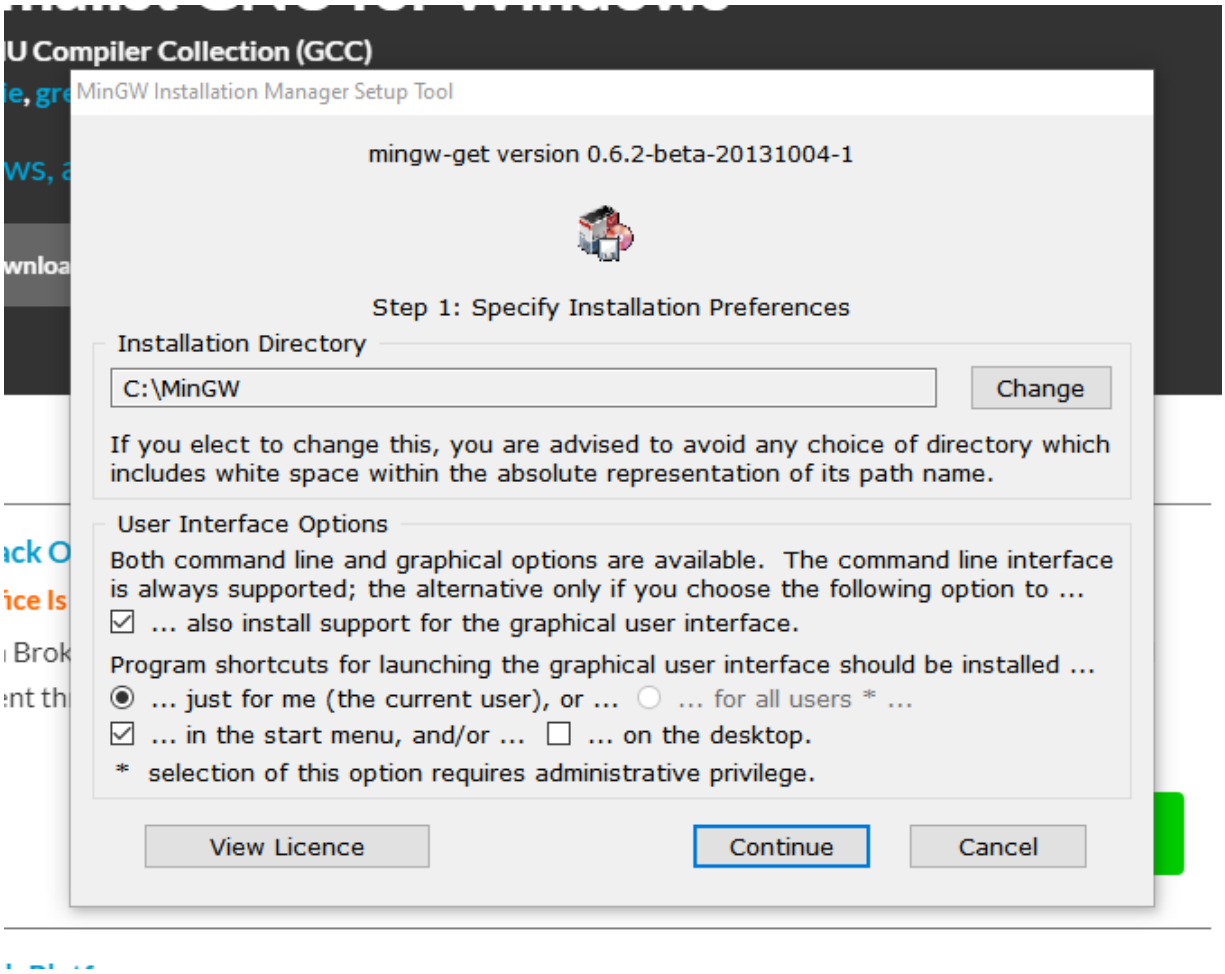
Download cmake from here. You should get an exe named similar to this: `mingw-get-setup (1).exe`

Install `MinGW / g++`

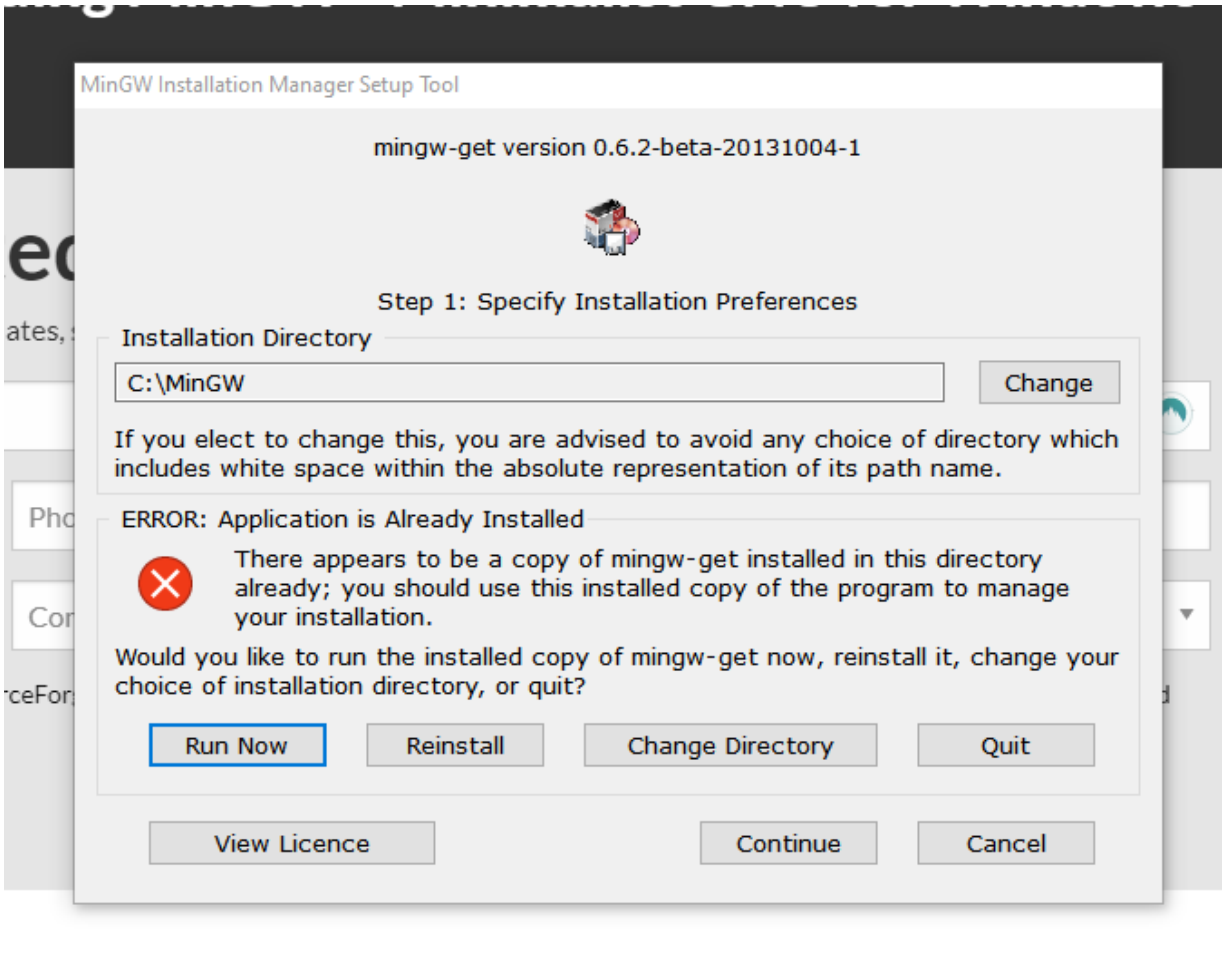
Open/run the executable, and follow the steps to install.



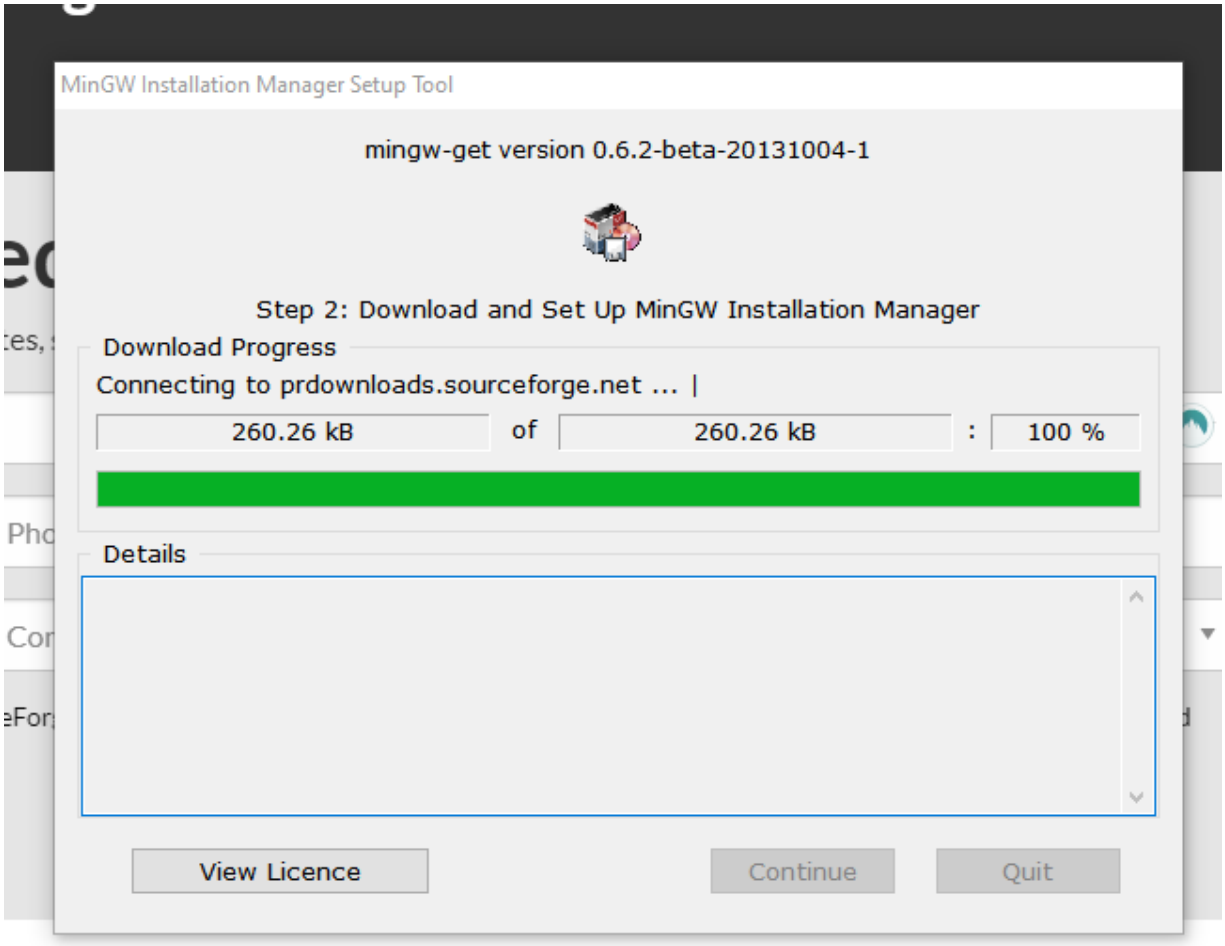
The default Installation location should be C:/MinGW. Once again, you can add a Desktop shortcut, but it will not be neccesary for this class.



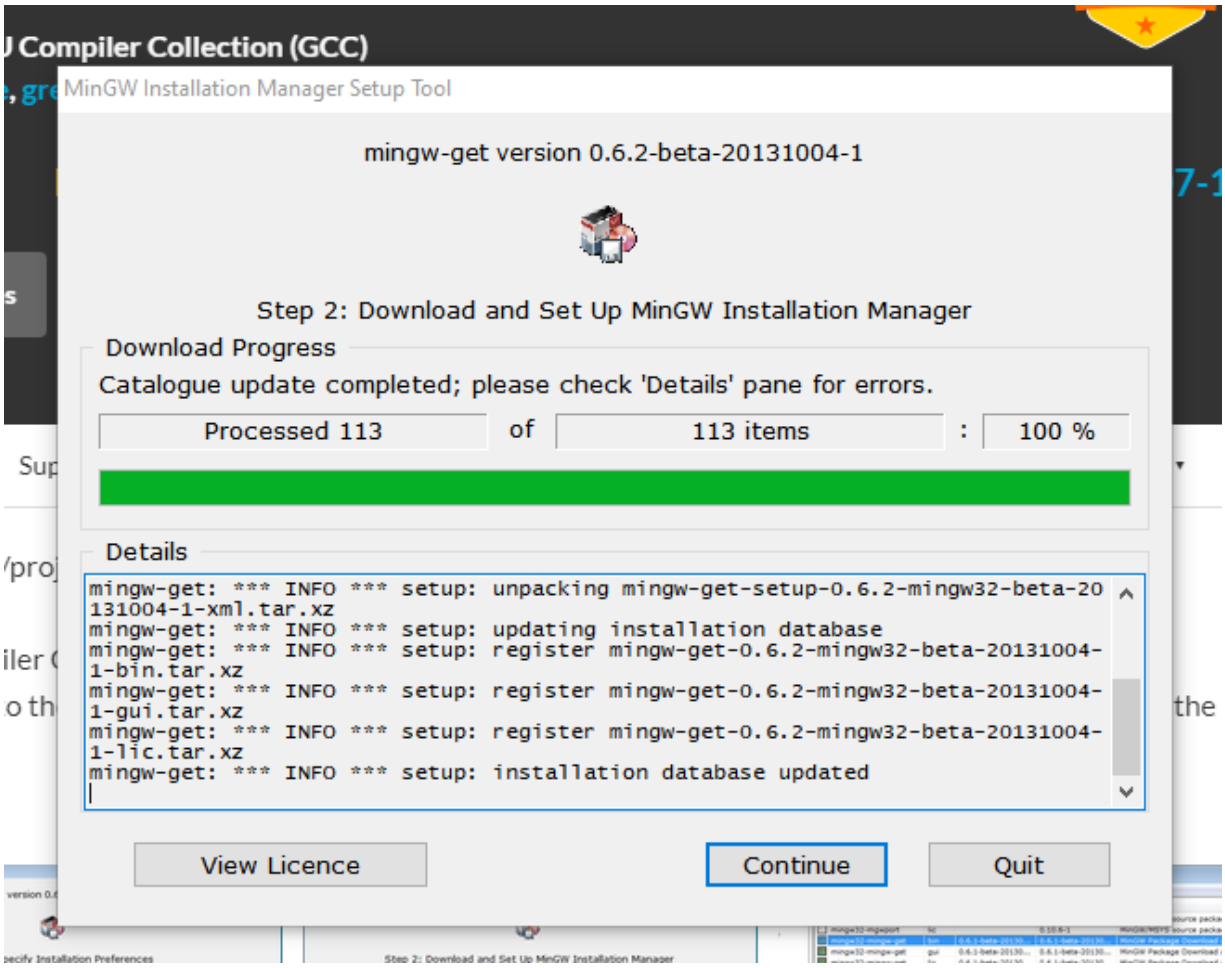
If you get a warning that MinGW is already installed, you can either reinstall and continue with this walkthrough or choose Run Now and skip to the steps below with the MinGW Install Manager.



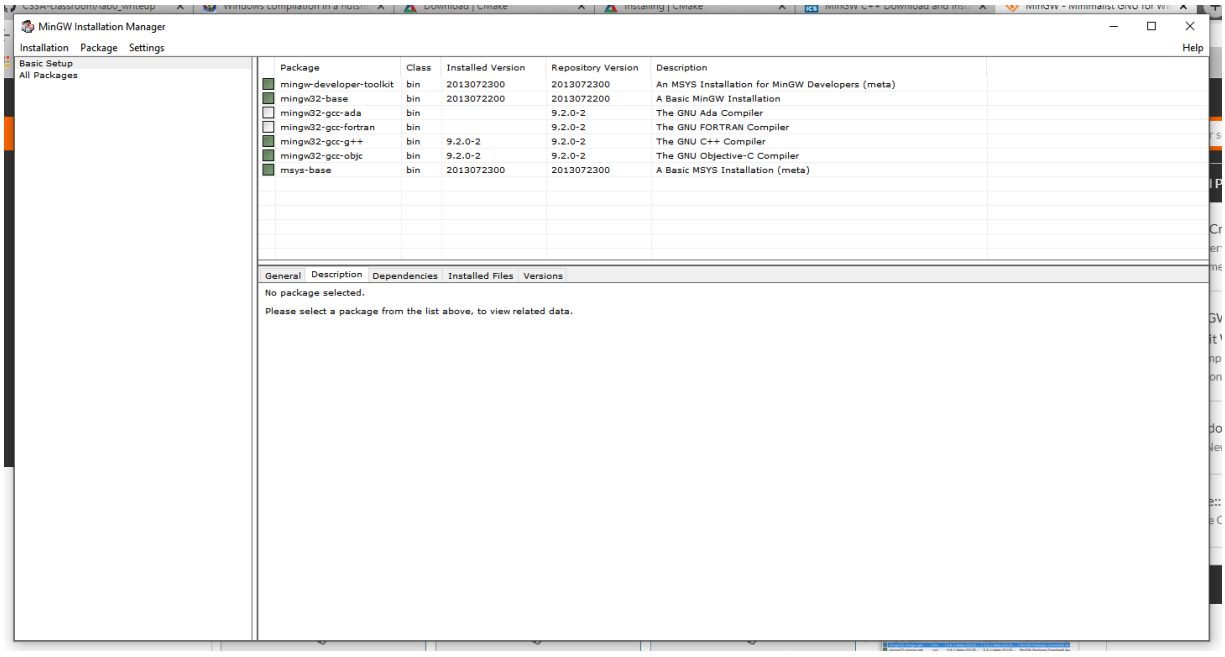
Installing..



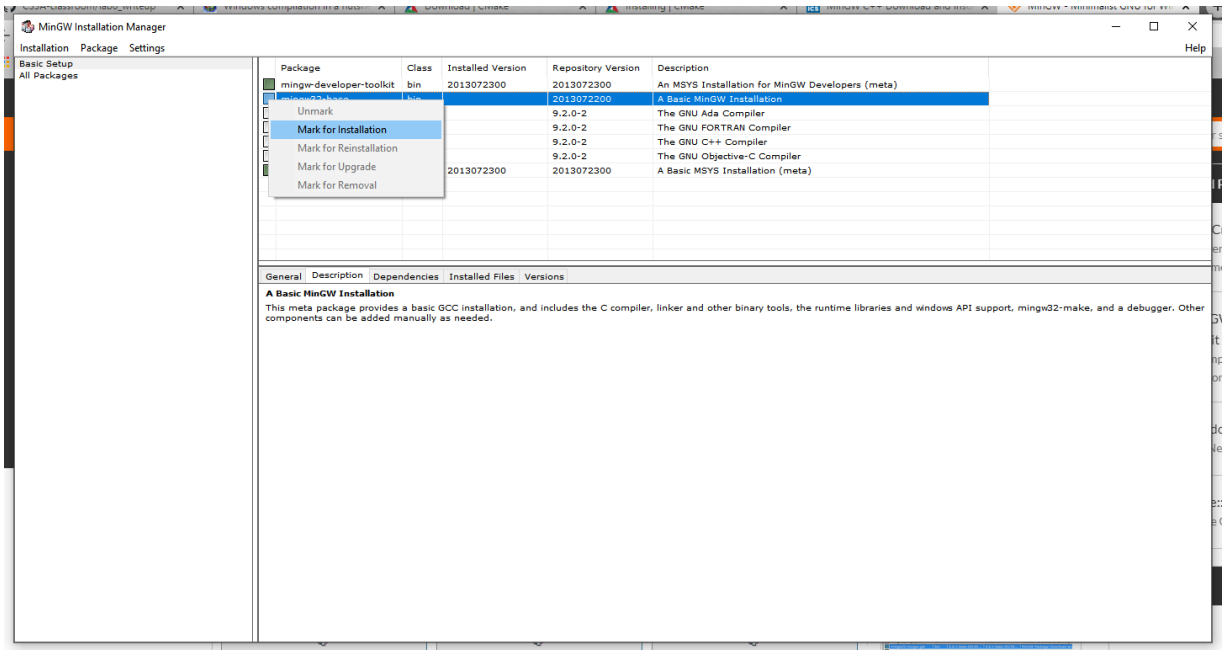
When it is done installing, click Continue.



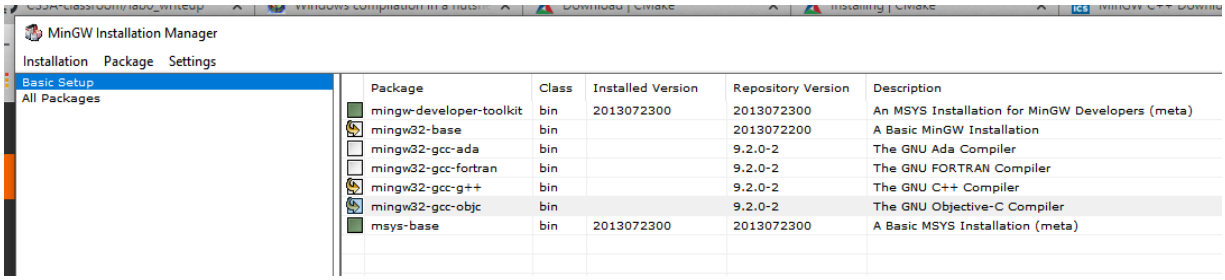
The **MinGW Installation Manager** should be opened up automatically. For this class, you will need mingw32-base, mingw-gcc-g++, and mingw23-gcc-objc.



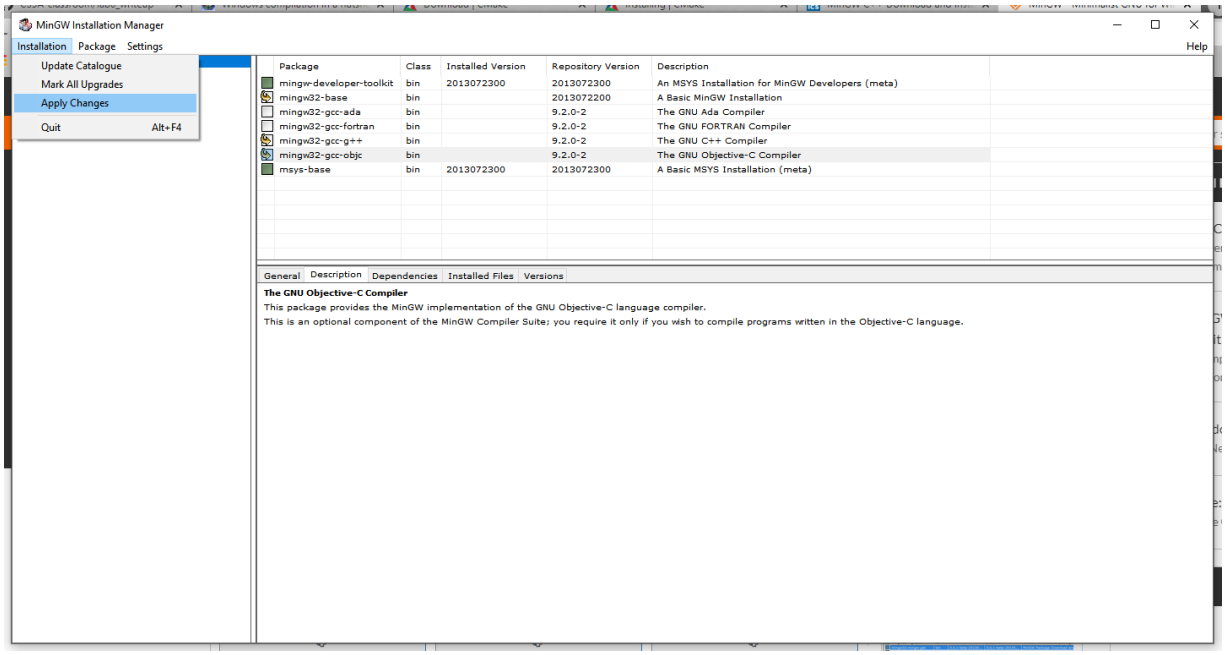
To select a package for installation, right click on it in the menu and select Mark for Installation.



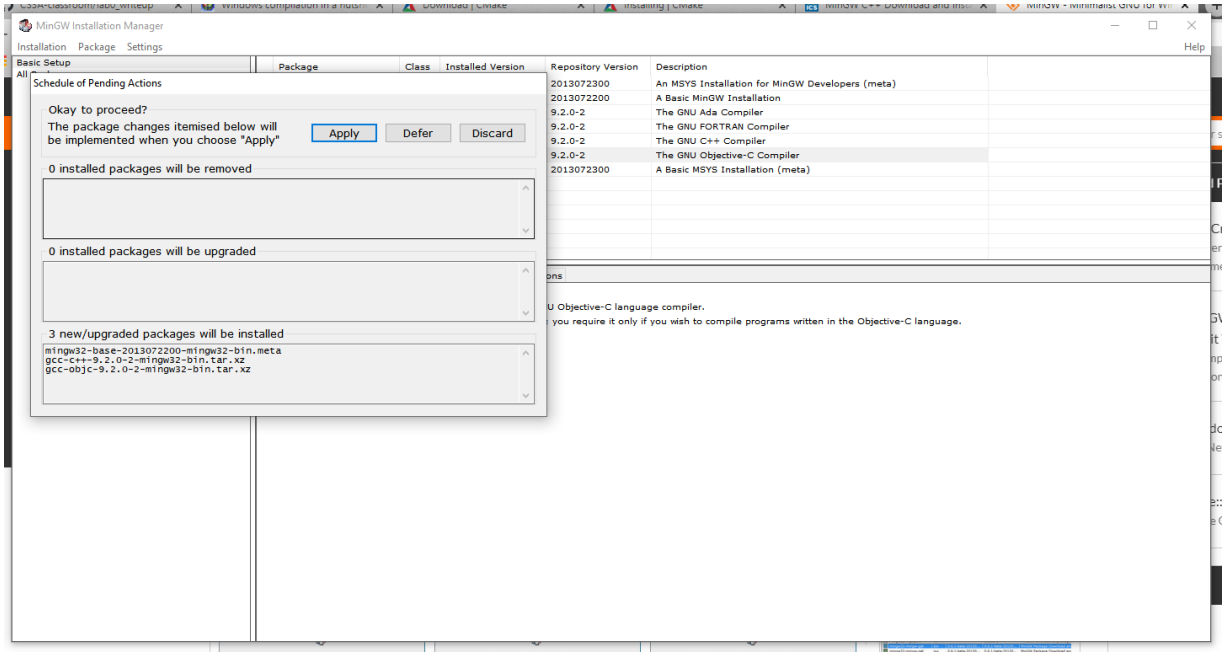
Marked packages will be selected like this:



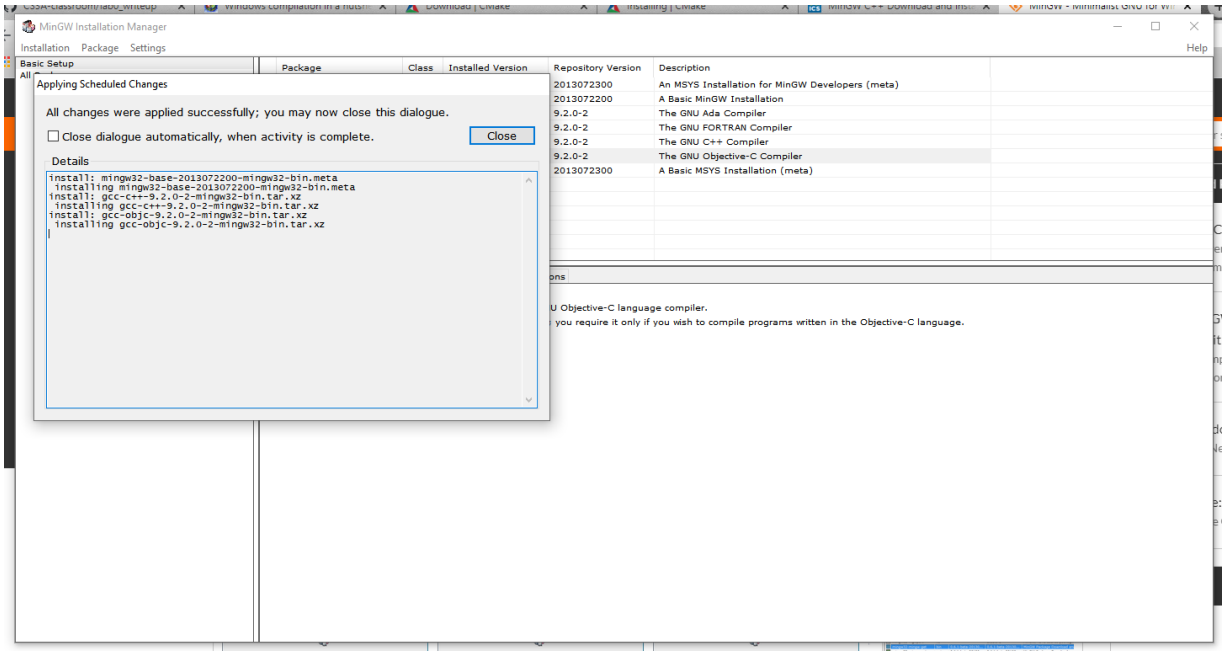
Once all neccesary packages have been marked, go to Installations > Apply Changes to install.



Click Apply

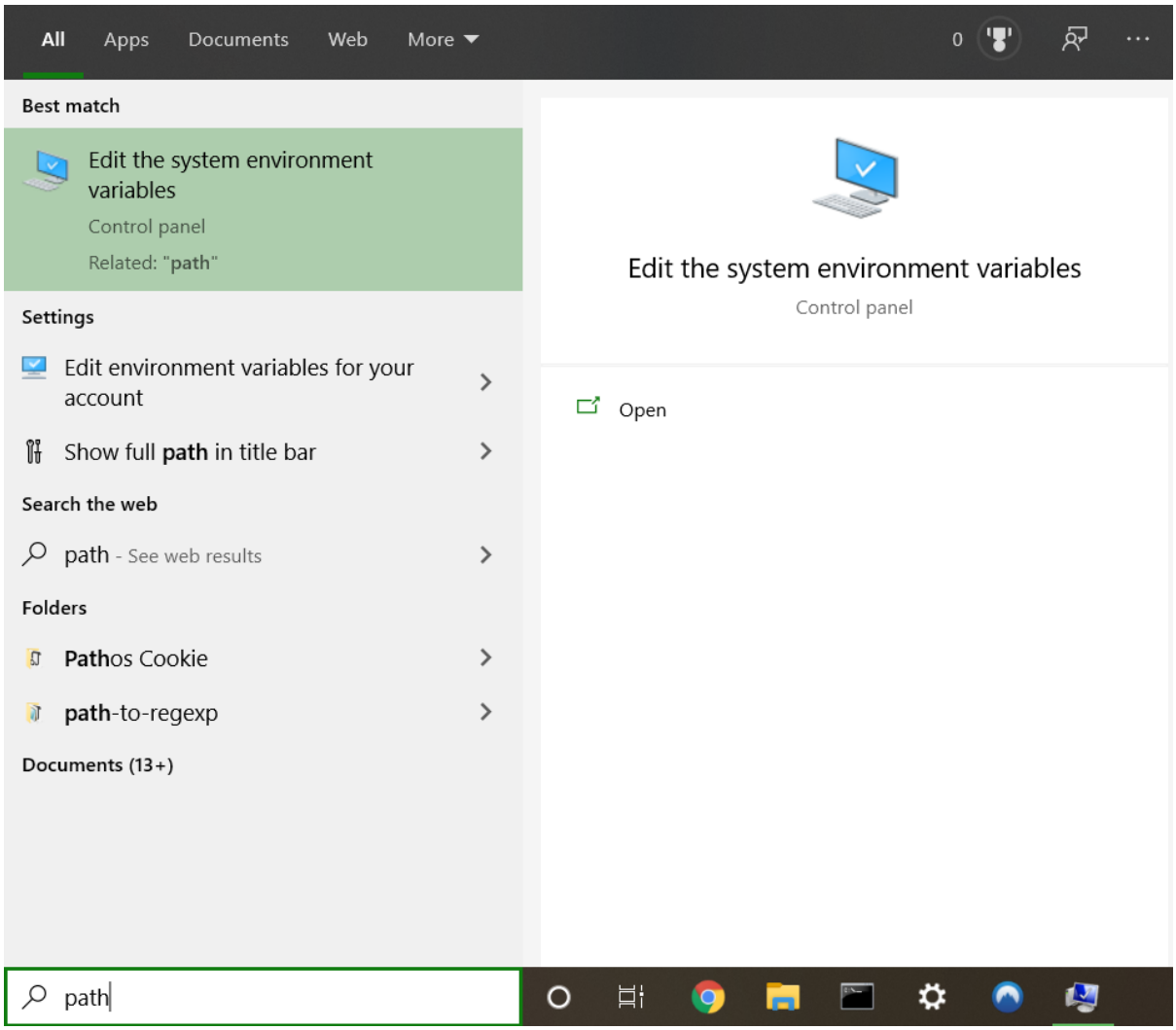


Once the changes are applied you may close the Installation Manager.

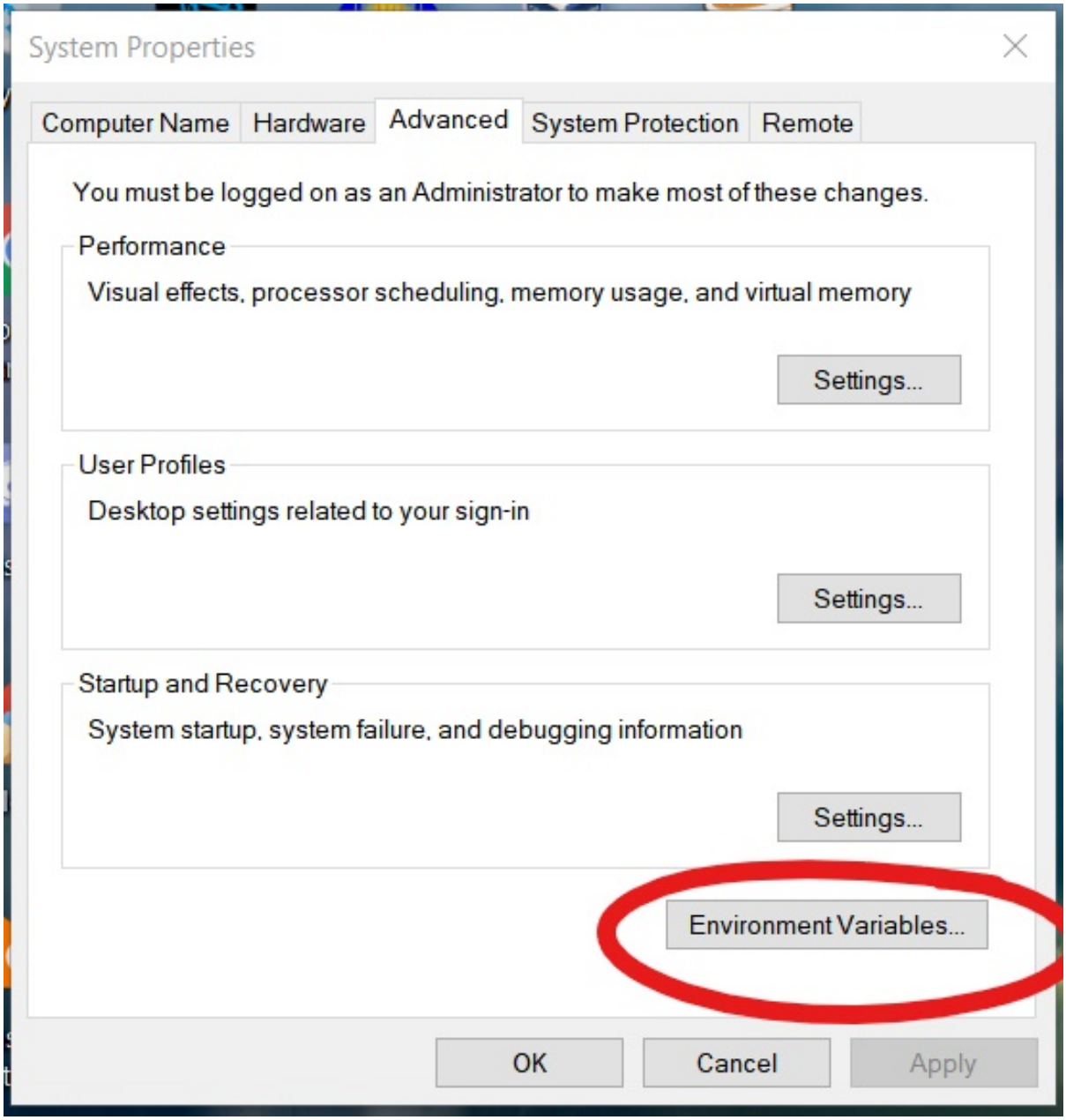


Add g++ as a System Variable and to your path

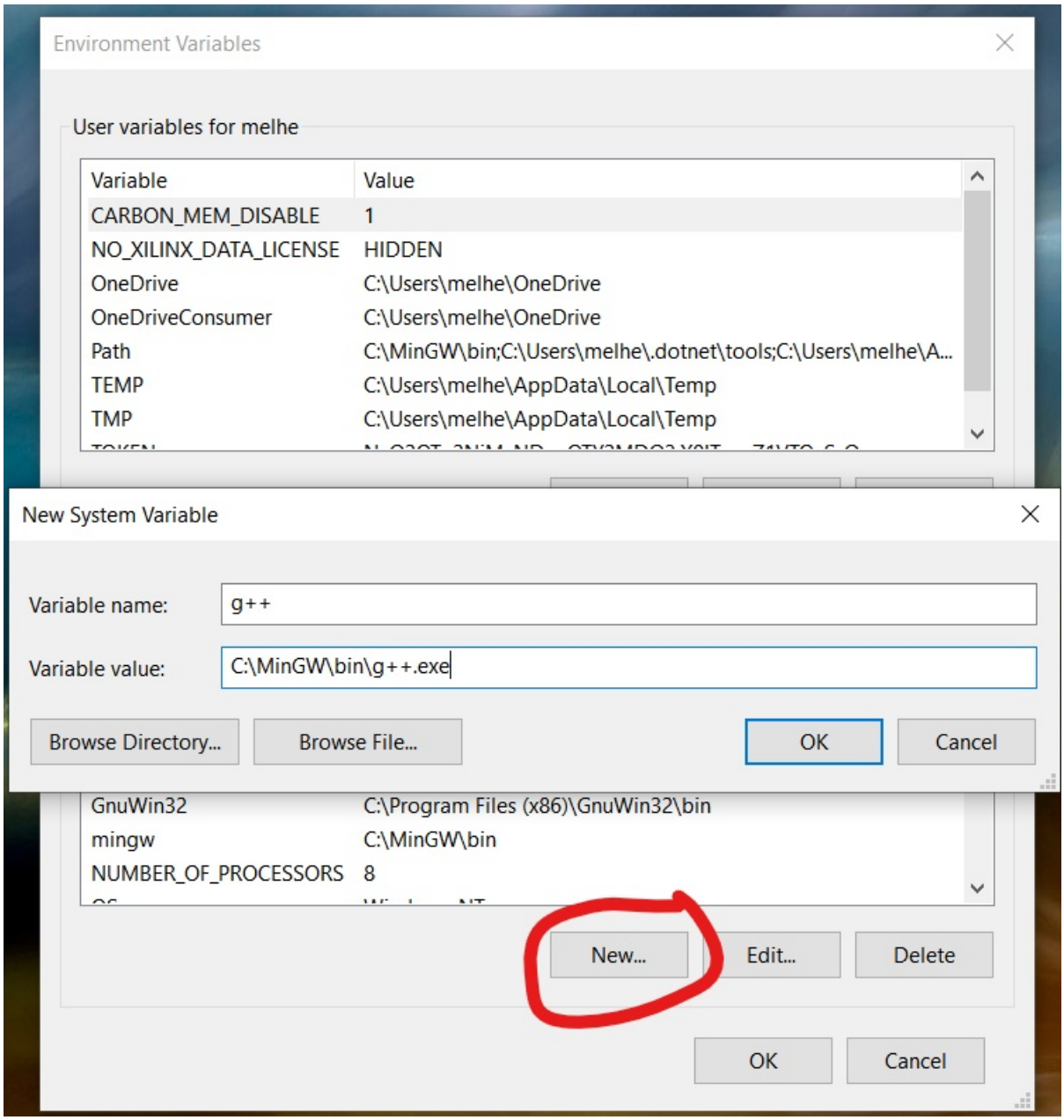
Search for "path" in the task bar search box. Open Edit the system environment variables.



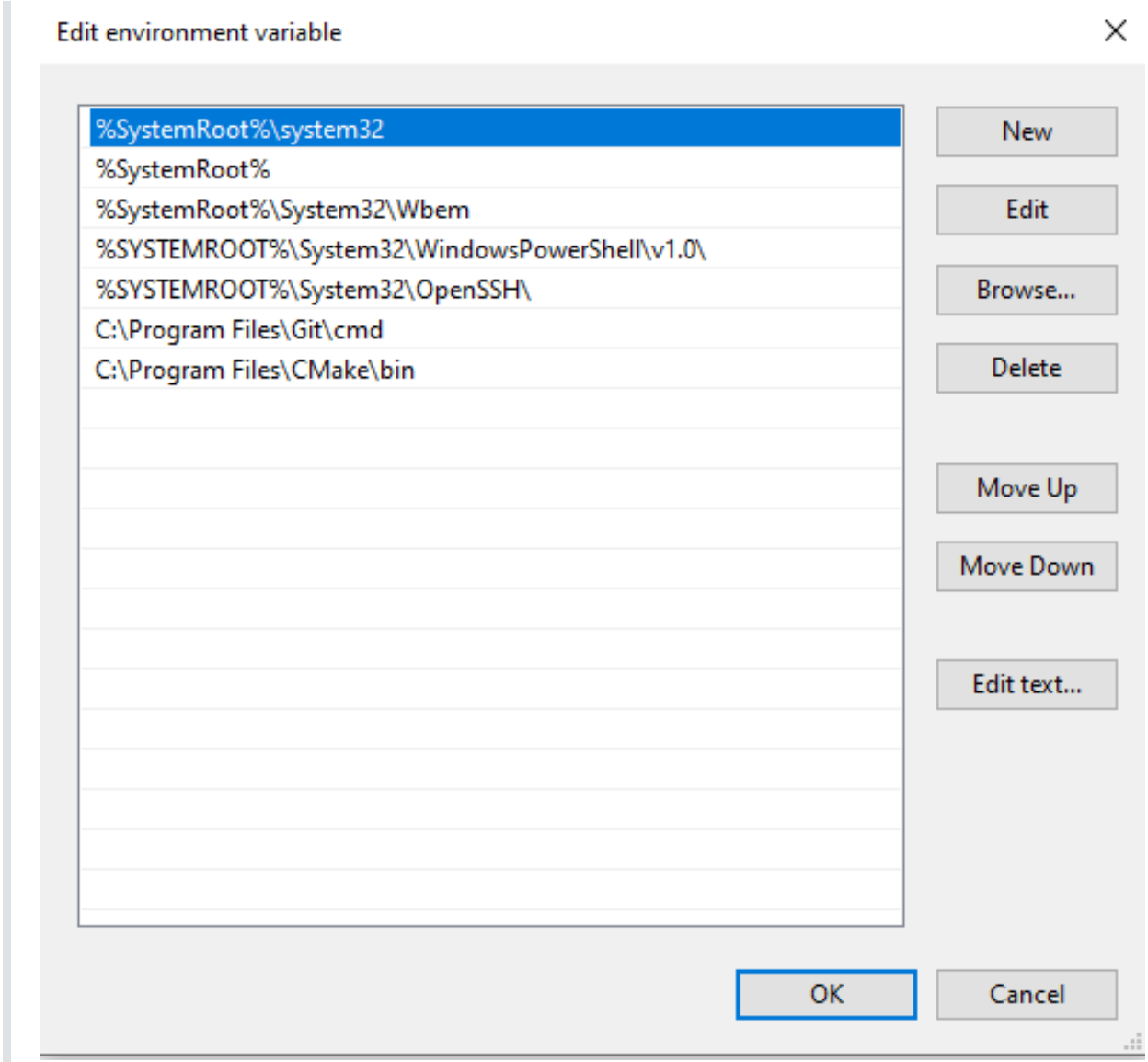
Click on "Environment Variables..."



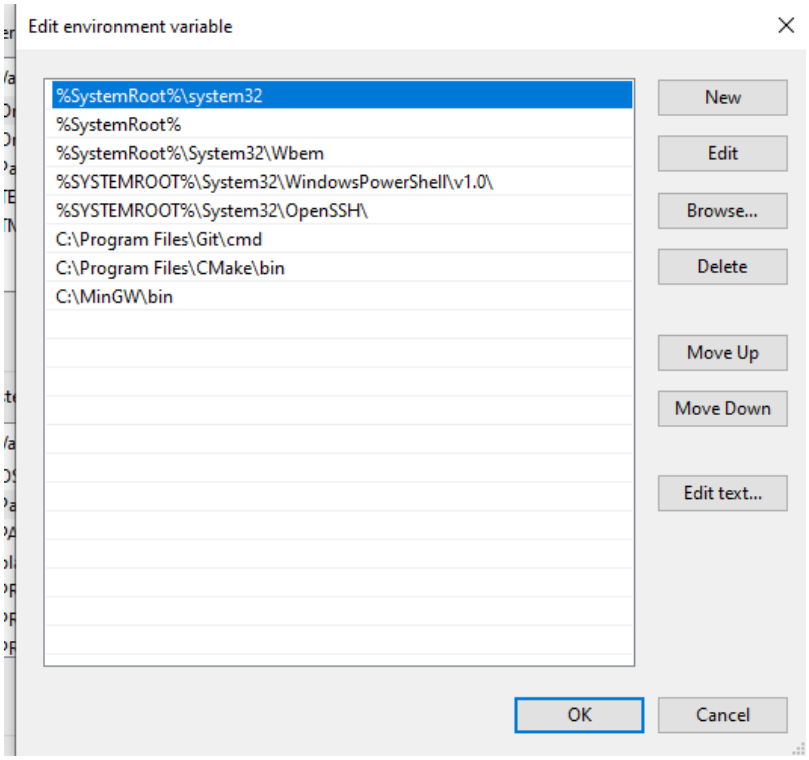
Add the path to the gcc and g++ executables to environment variables as shown below and press OK.



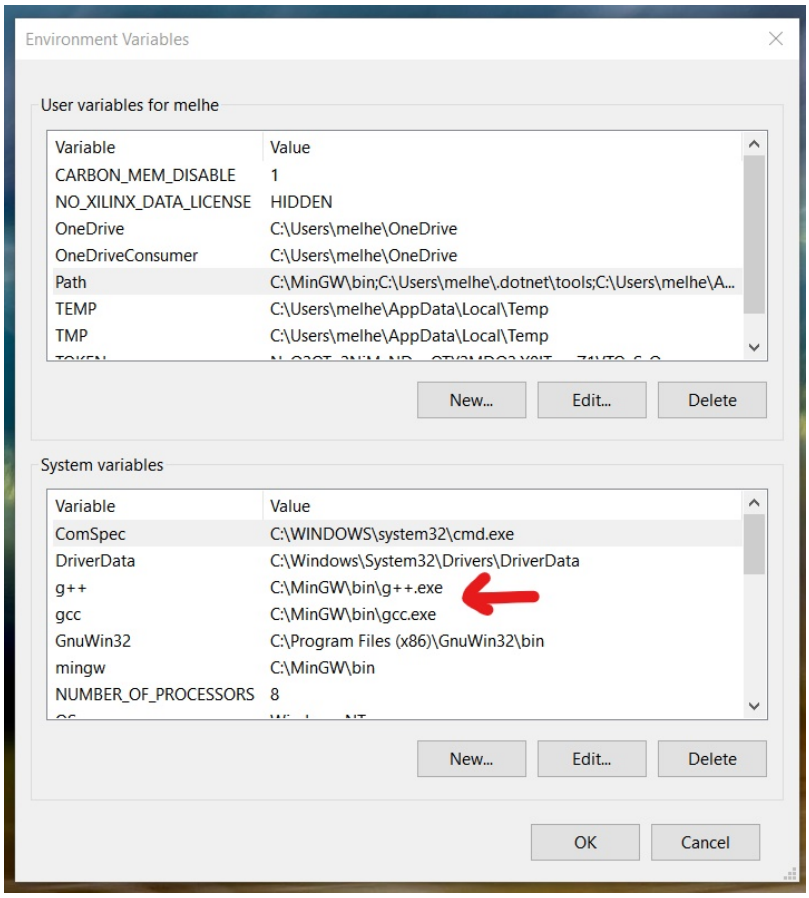
Next, double click on the **user variable** Path (in the top half of the window). A window like this should pop up:



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

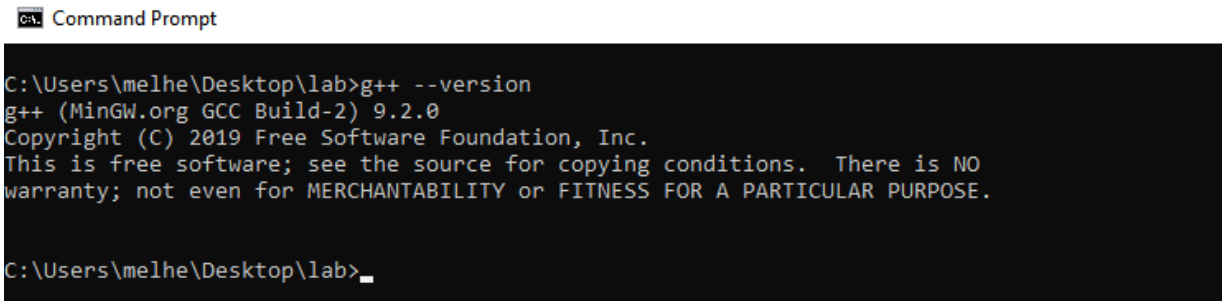


Your environment and user variables should look like this after you are done:



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.

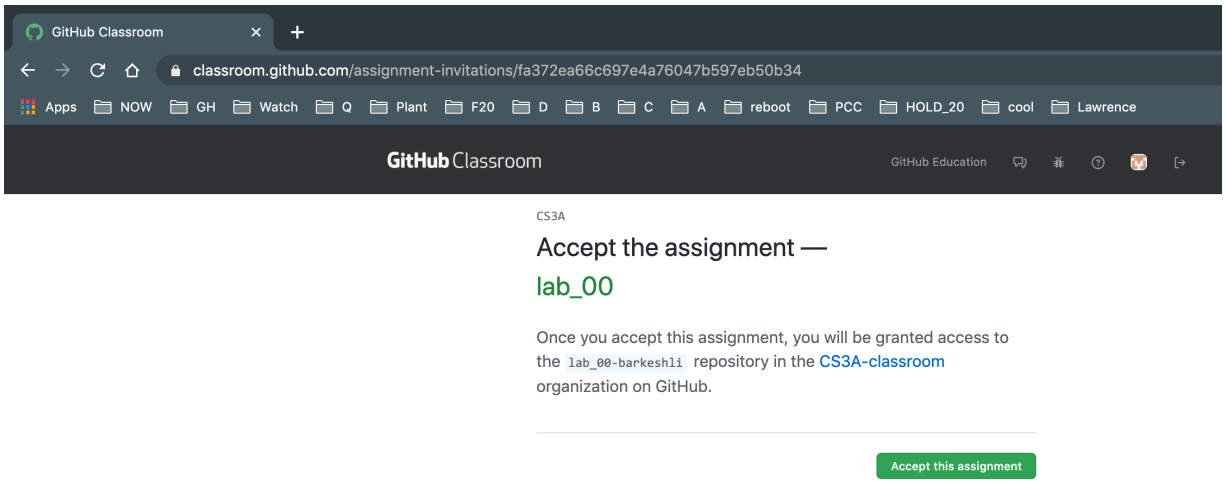


Accept the assignment

Here is the assignment link for CS3A and here is the link for CS8

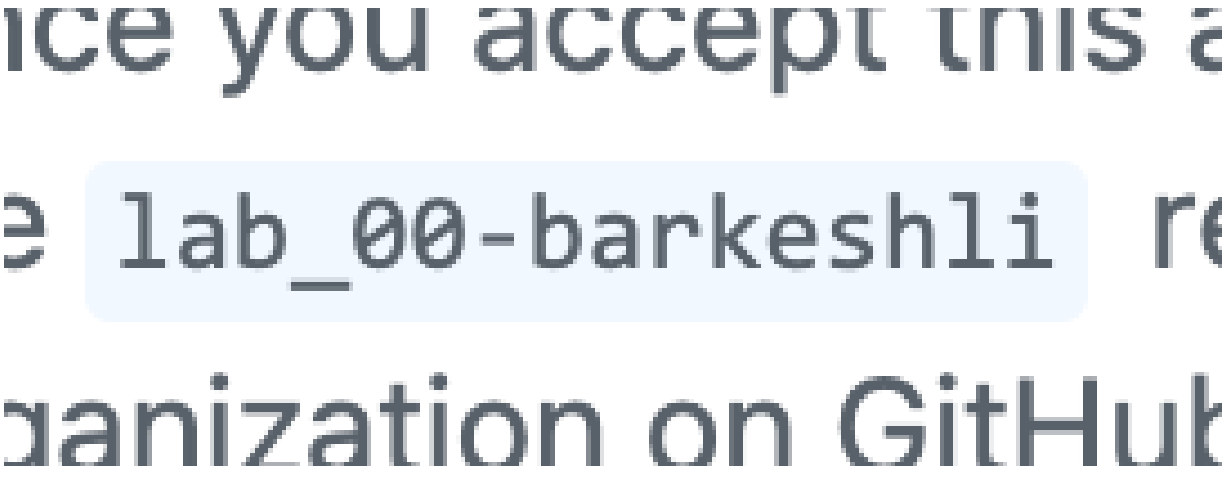
Accept assignment page:

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.



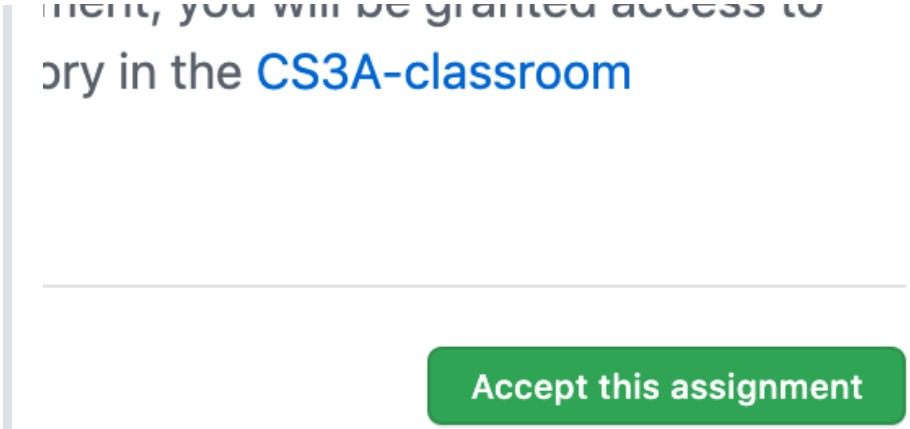
Your repo name:

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



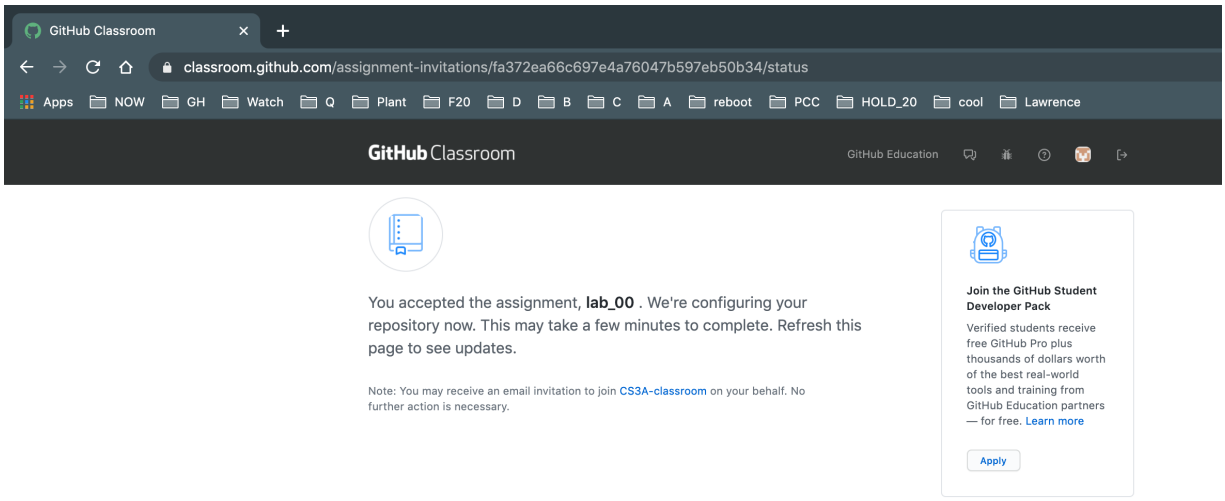
Accept the assignment

You will *accept* the assignment by clicking the green button.



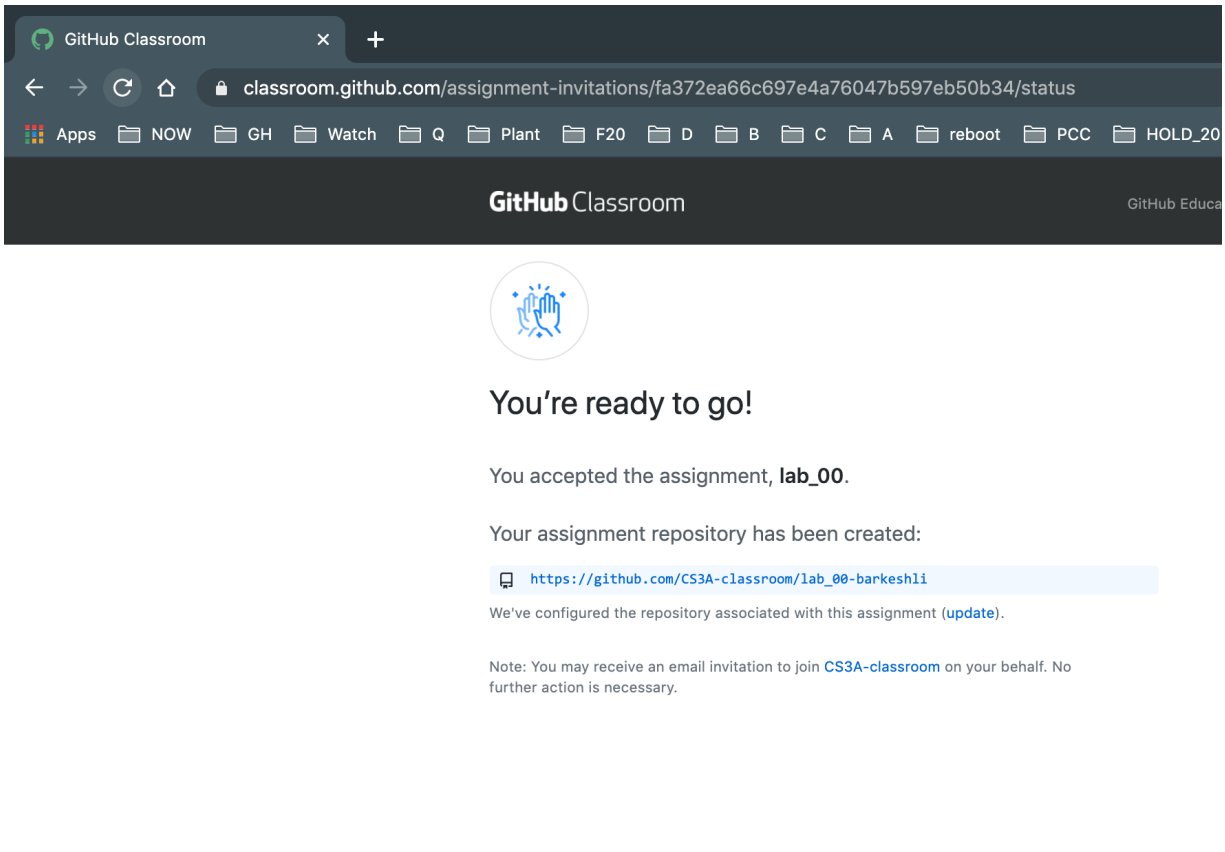
Your assignment repo being created.

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



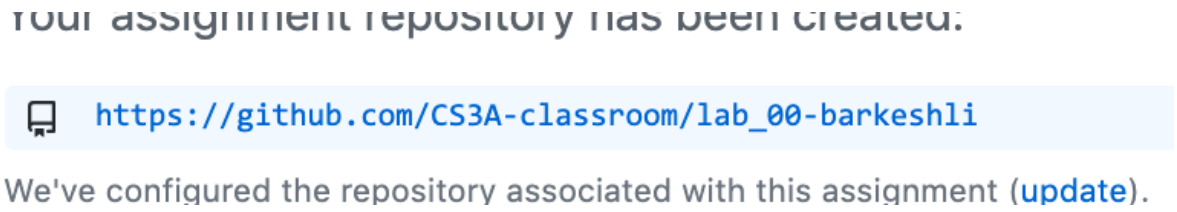
Your assignment repo:

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



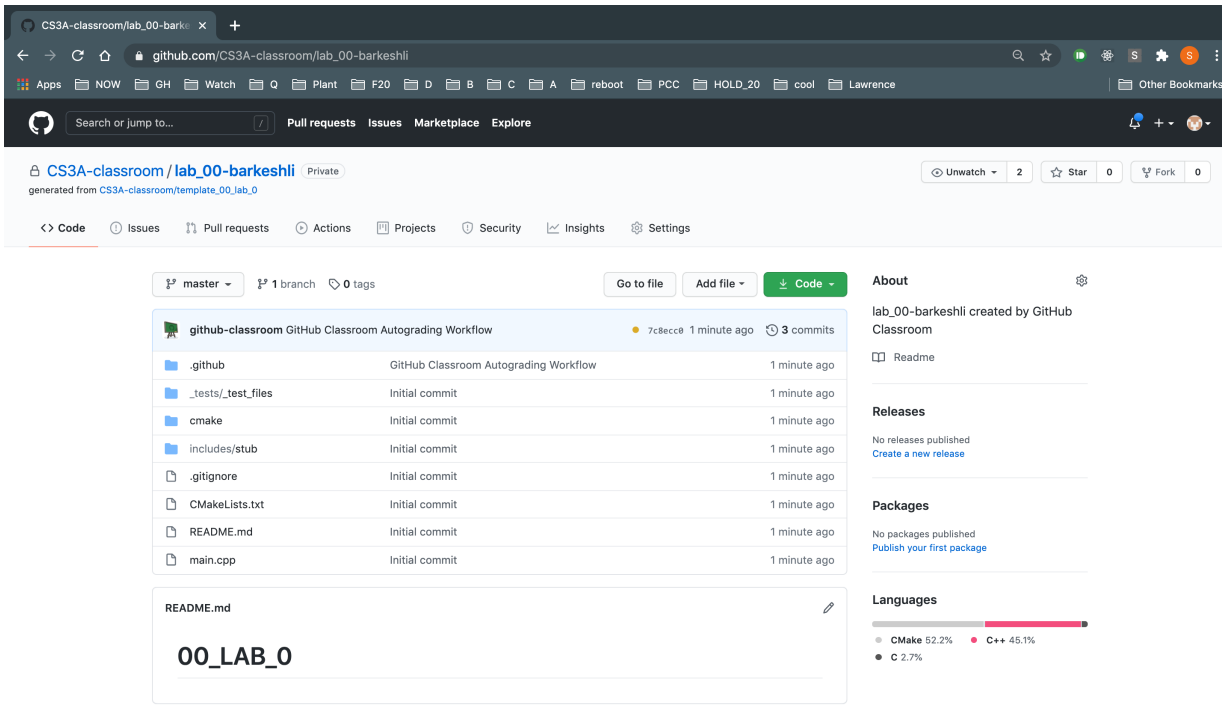
look closer:

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



Your assignment repo:

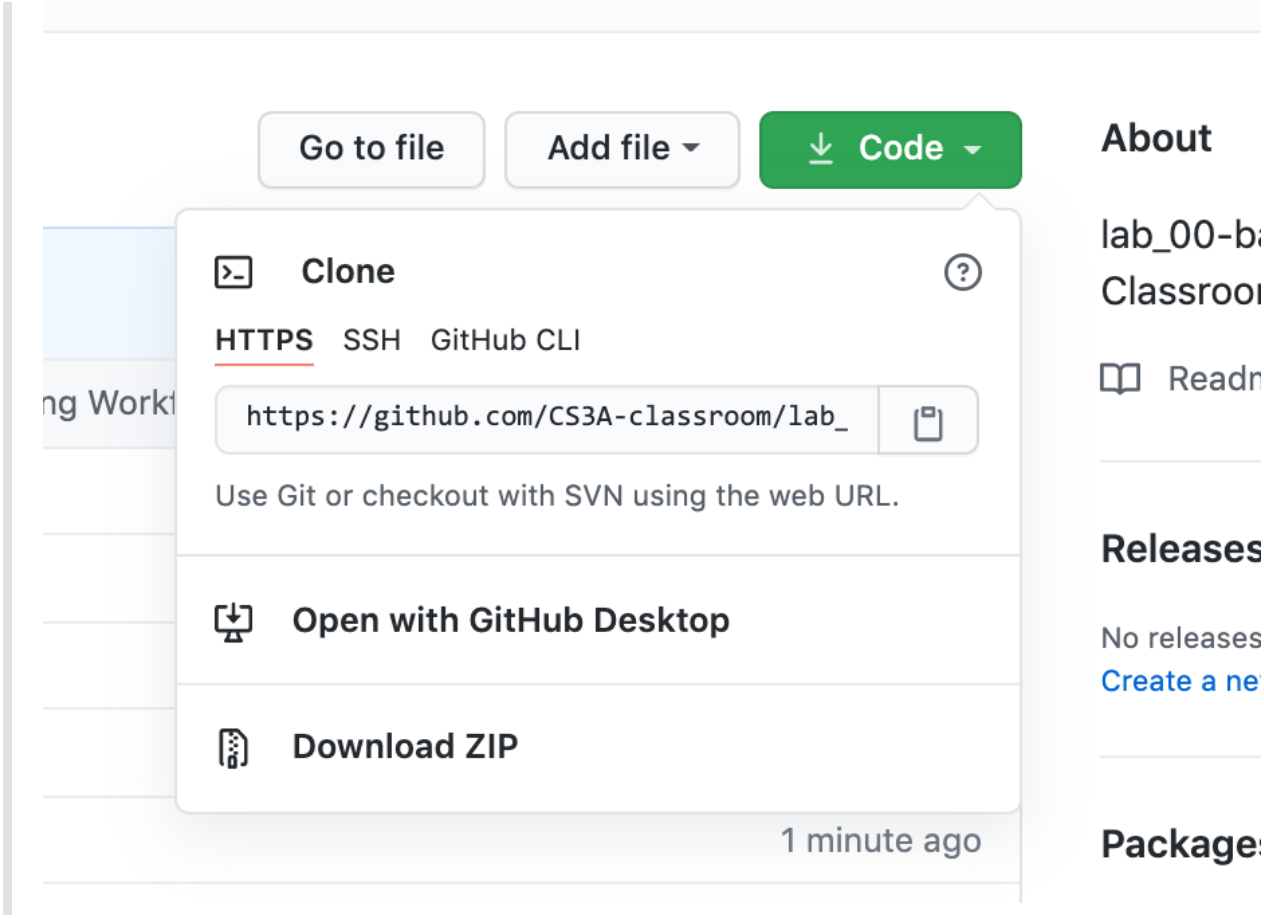
Bookmark this page or know how to get here. We'll need to check in here soon.



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:



clone the assignment repo:

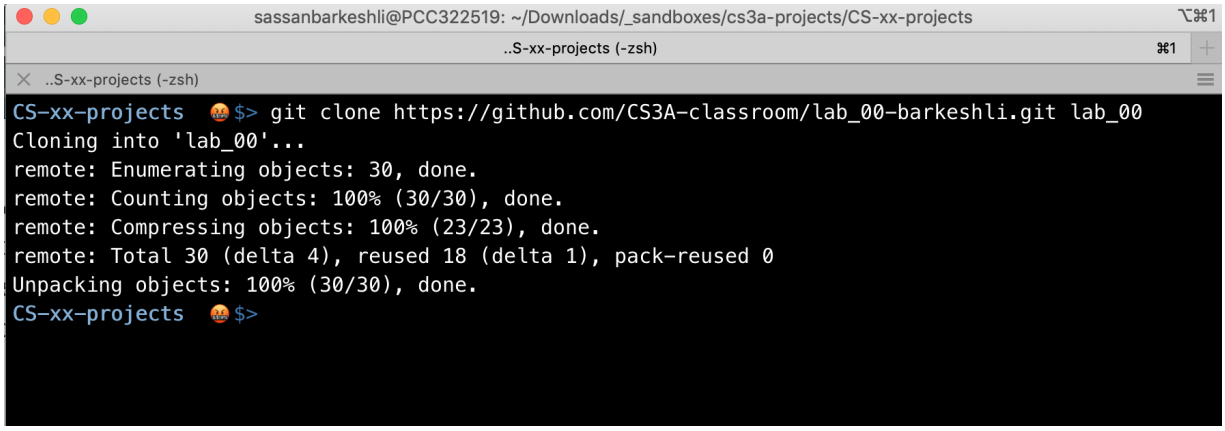
Before you can work on your project, you will need a local copy of the assignment. This is called **cloning** the repository.

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

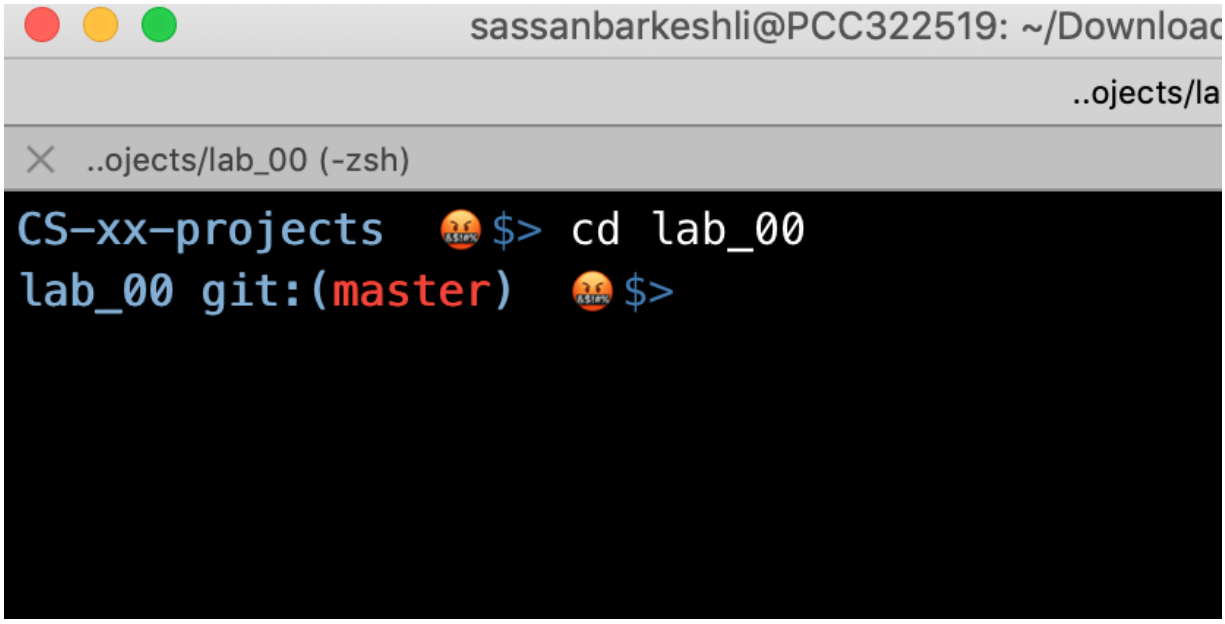
`cd` into your projects folder and clone your project there.

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`



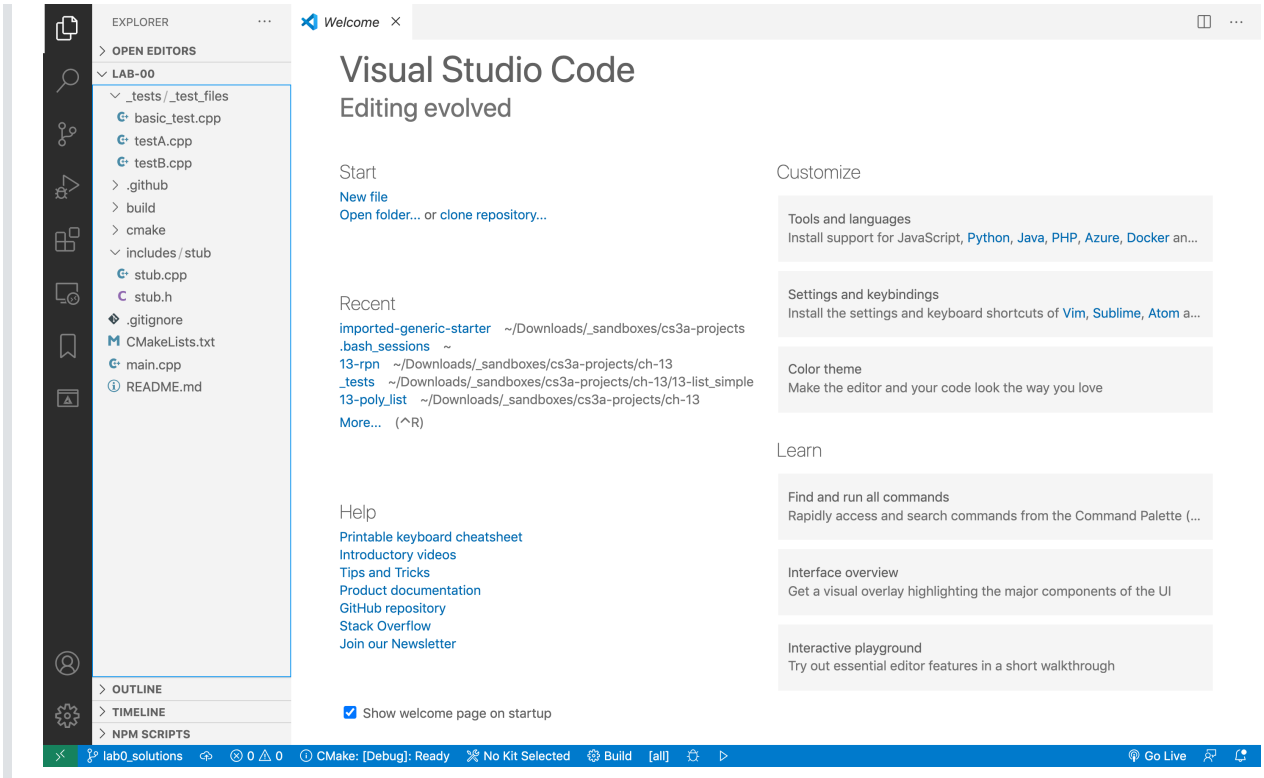
`cd` into this project folder:



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:

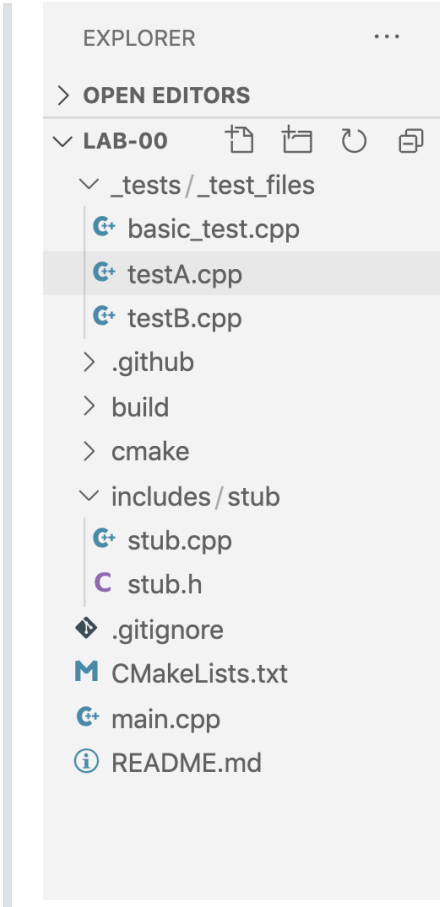
All the projects in this class will follow the same file organization.

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.



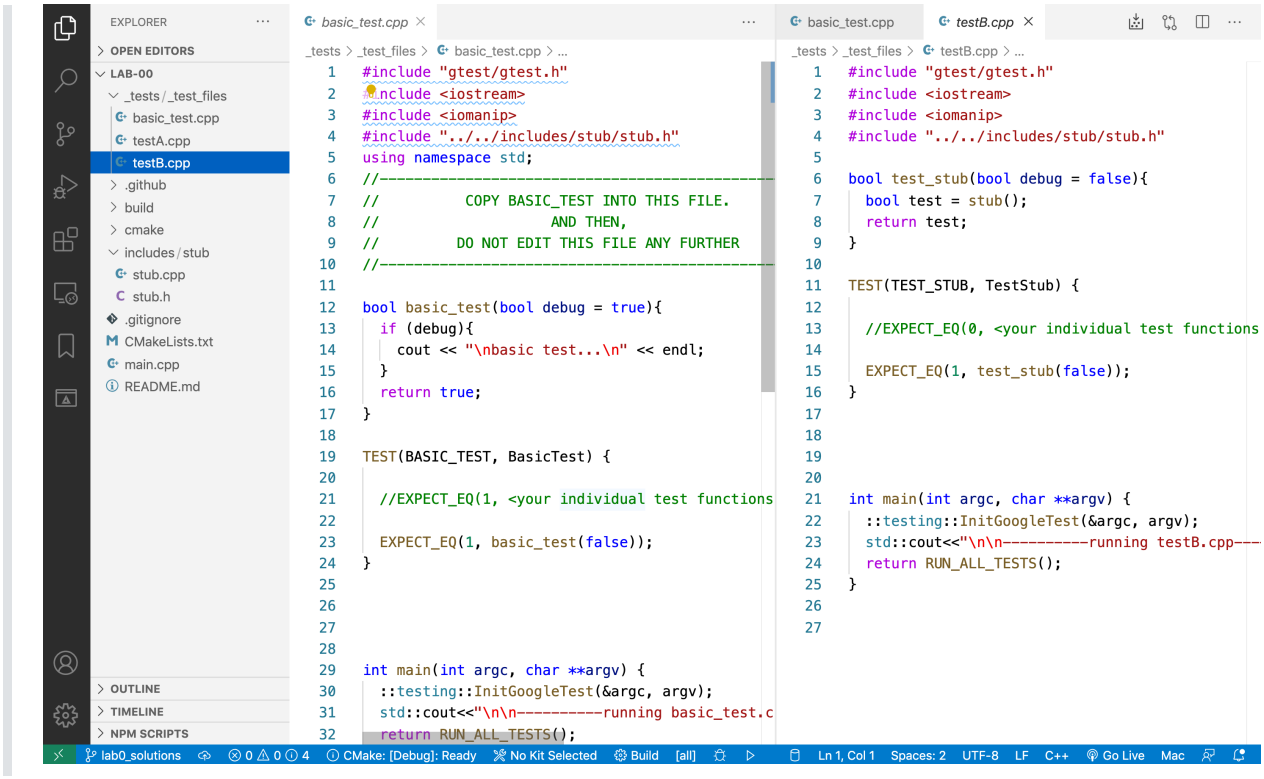
A look at the test files:

basic_test.cpp: My sample test goes here.

This is the placeholder file for a sample test file you wilkl 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 fils that will demonstrate the correctness of your poject are housed here. Part of your grade relies on the quality and success of the tests in this file.



CMakeLists.txt

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

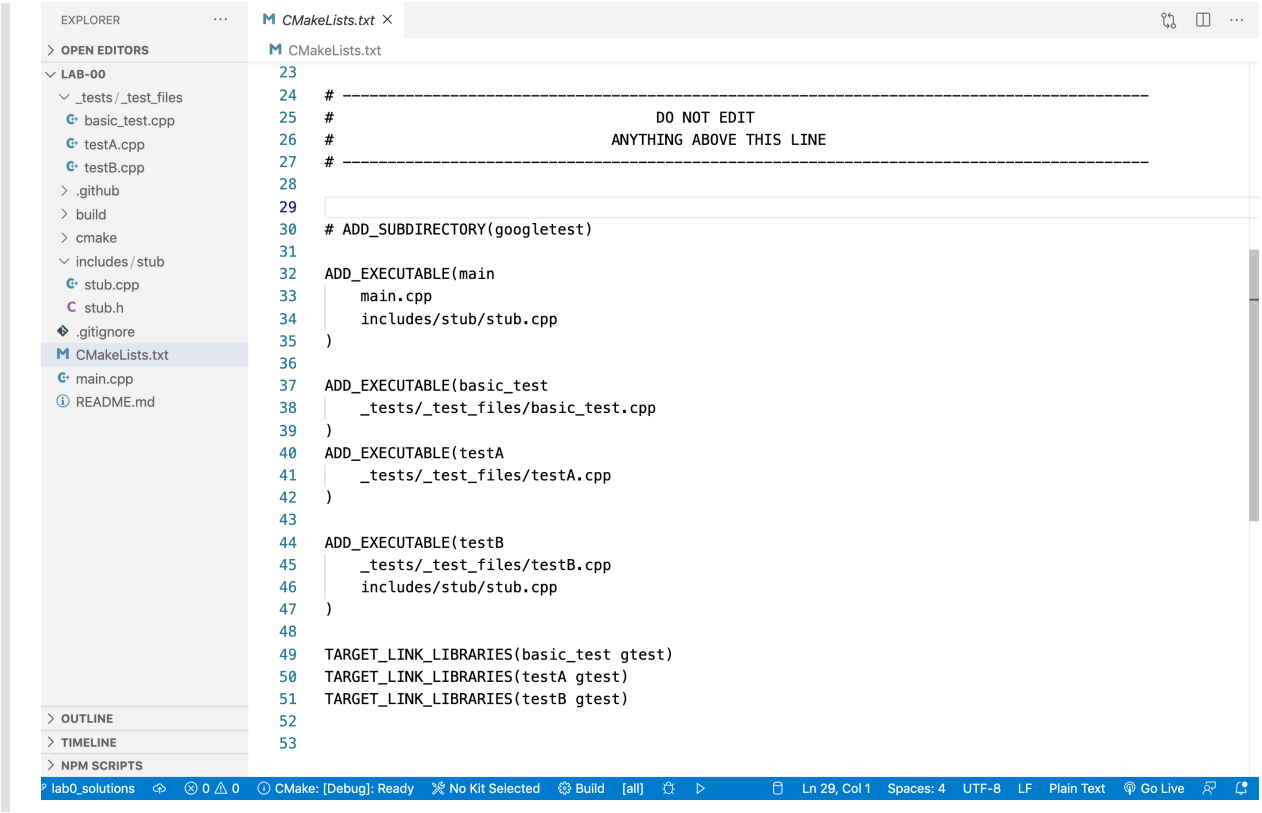
List your cpp files here

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.

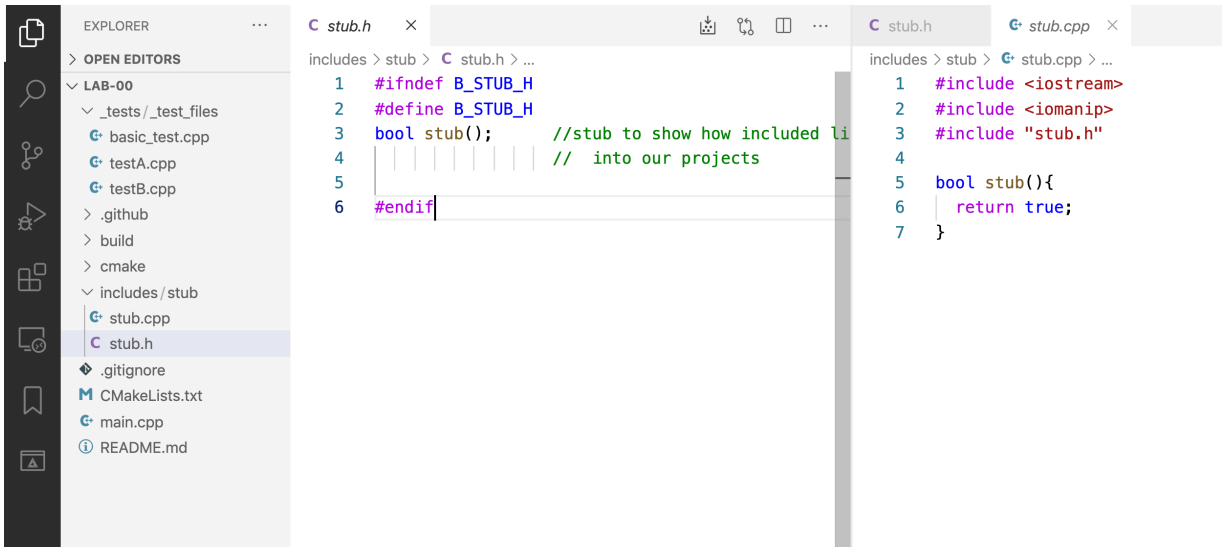
The grader will also use this file to build your project on the server side (once you submit - push your projects to github)

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...)



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.

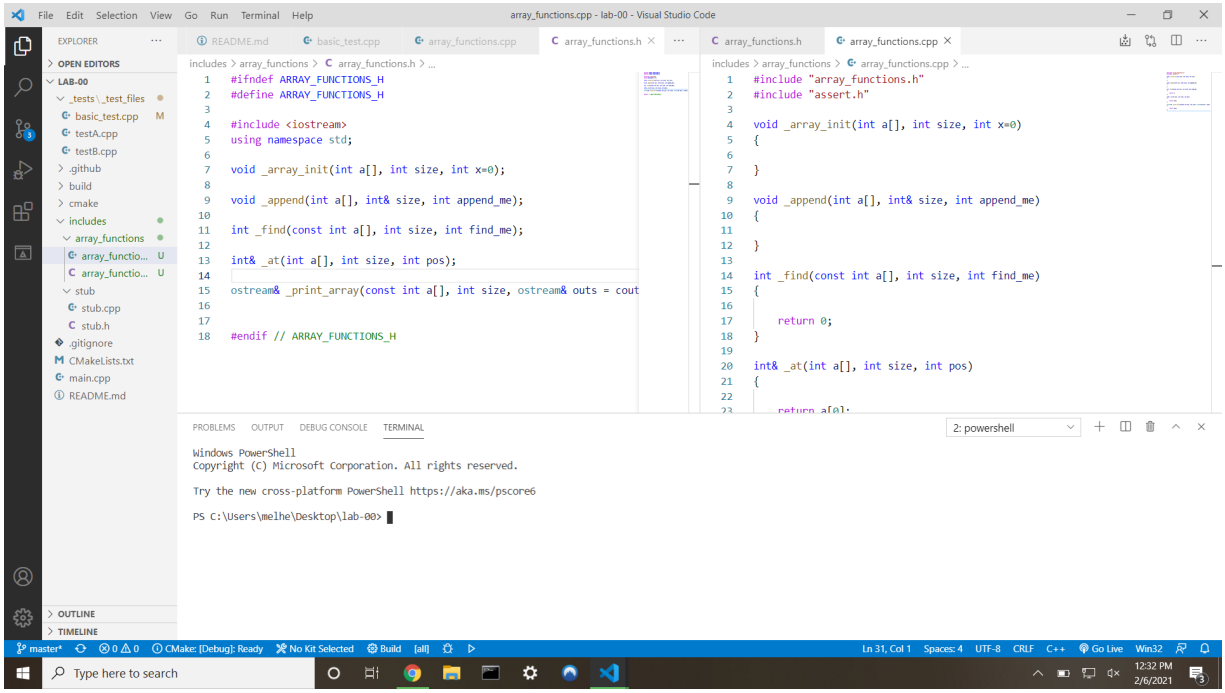


Open terminal:

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.



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

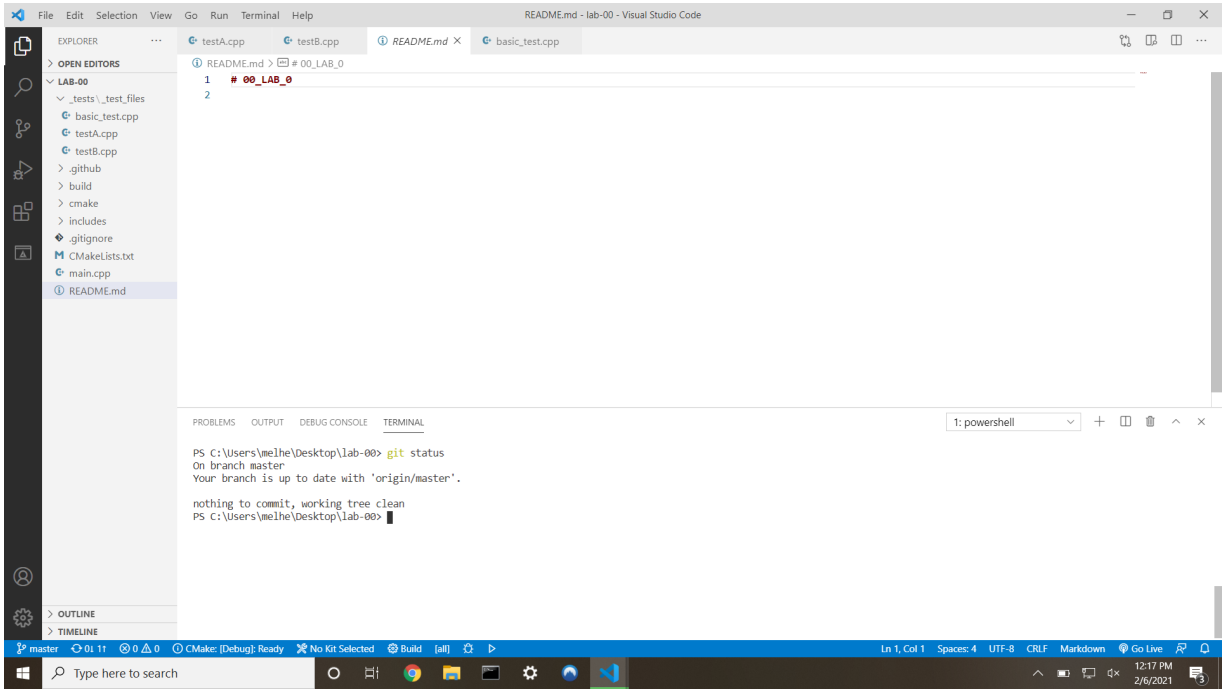
One of the main tasks in this class is tracking changes made to the project. We need to know what happened when and what changed. This is both for your peace of mind (helps you not lose your project accidentally,) and for me to track your progress throught he course.

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.

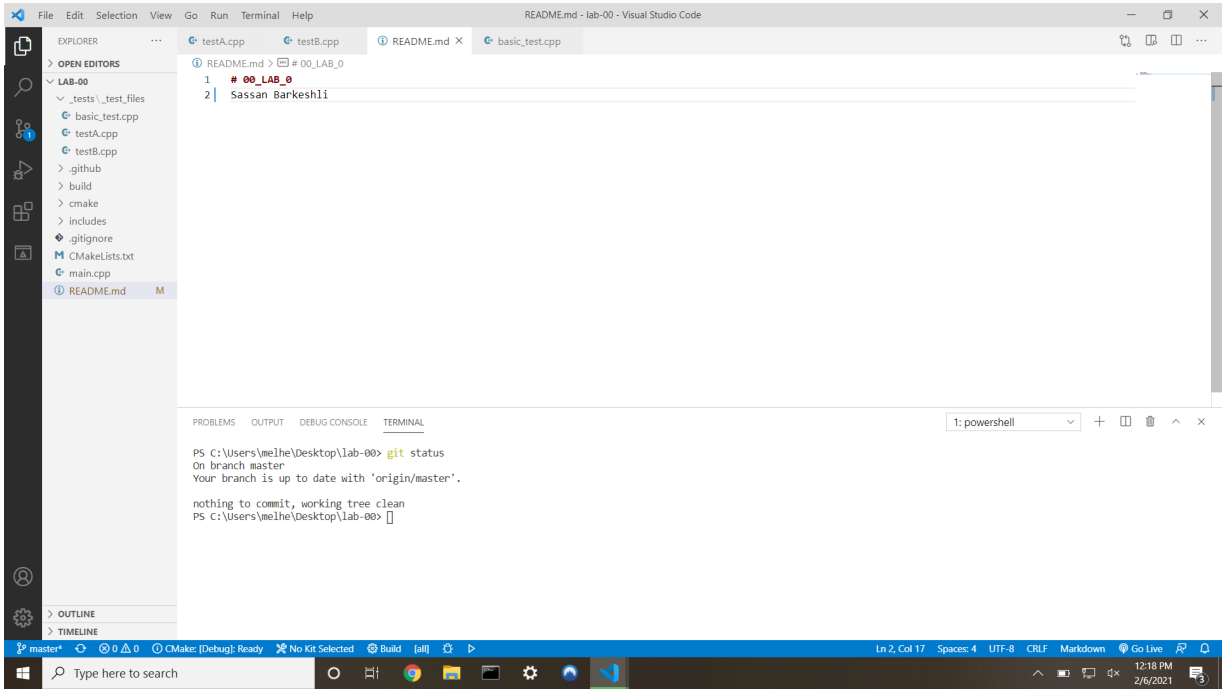
`git status`

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.



Edit the README.md

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



`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]

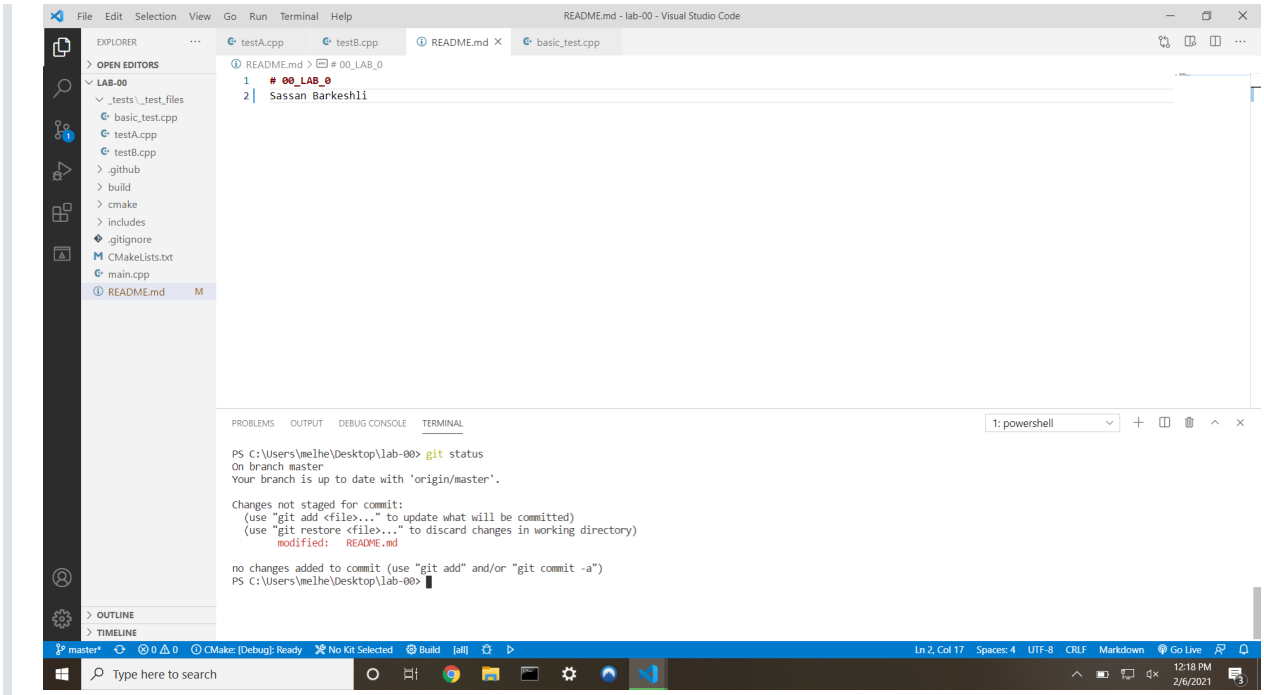
`---- git commit --->`

[committed changes]

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

[pushed, or synched with remote site]

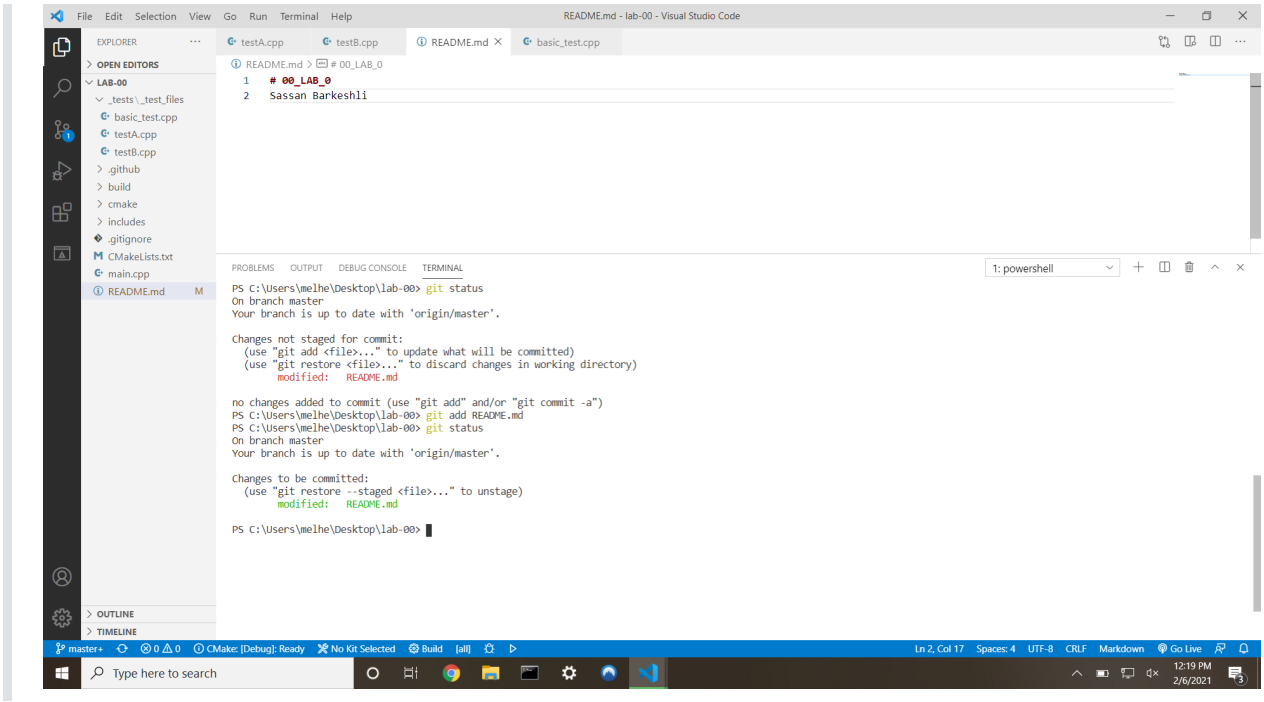
You go from zone to zone by the commands listed above.



`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*



`commit` your changes:

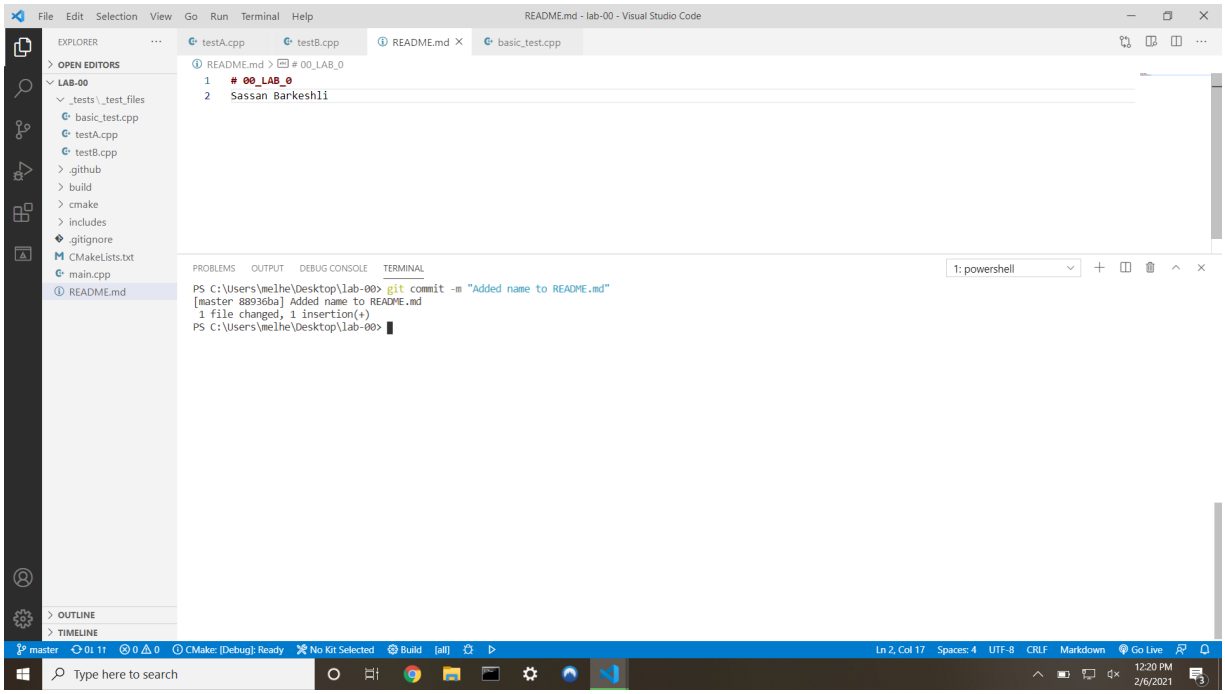
You `commit` your changes to permanently record these changes to `git`. If somehow you ruin your project (as we all have done from time to time,) you can *revert* to this state of your project.

`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.



Has Github assignment repo changed?

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

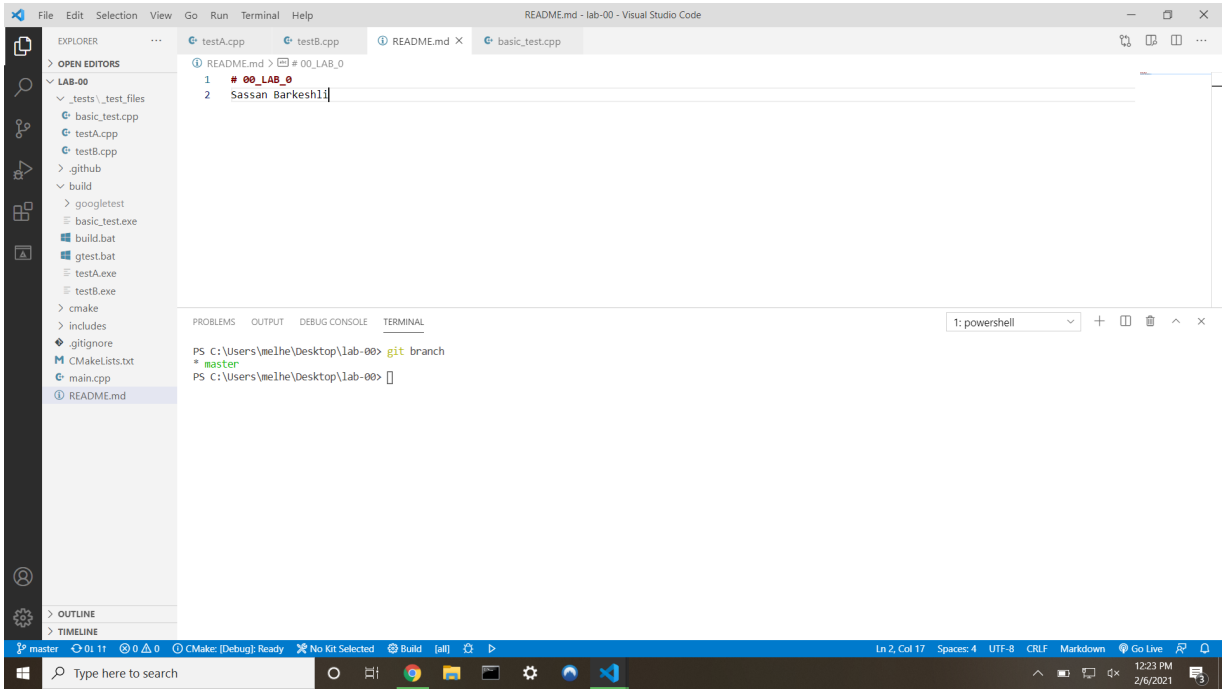
`commit` ting only records your changes on your local machine.

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:



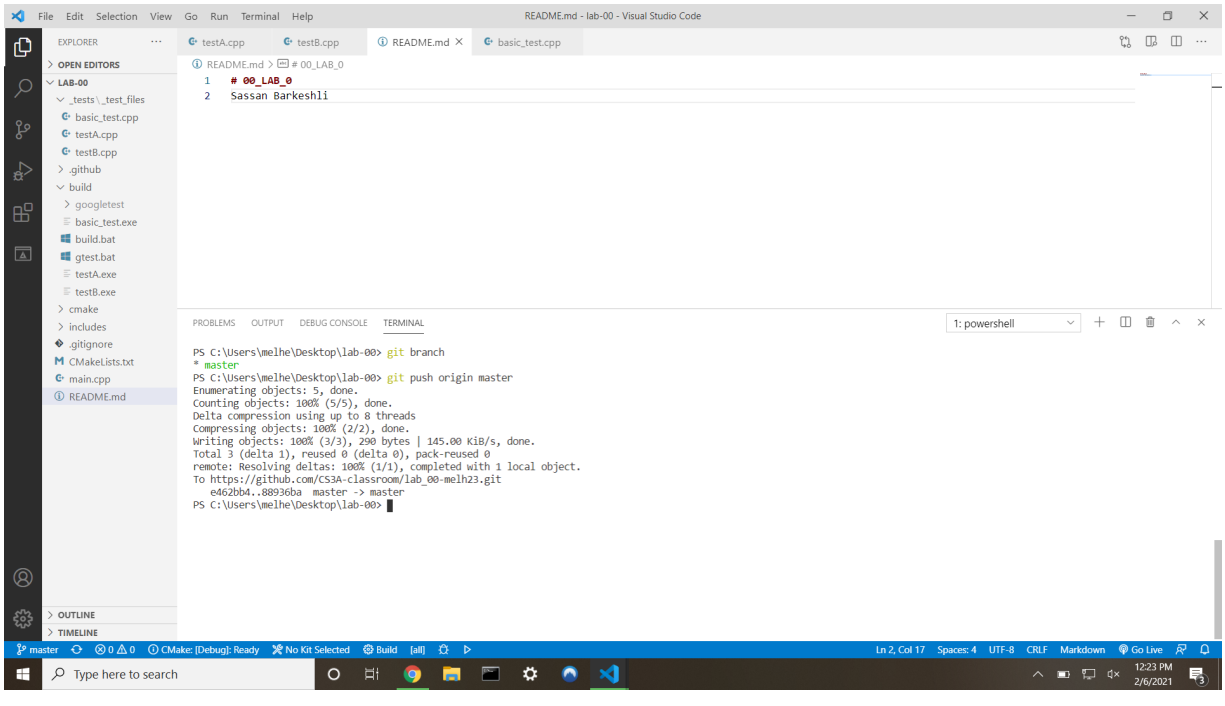
Zoomed in:

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

`git push` :

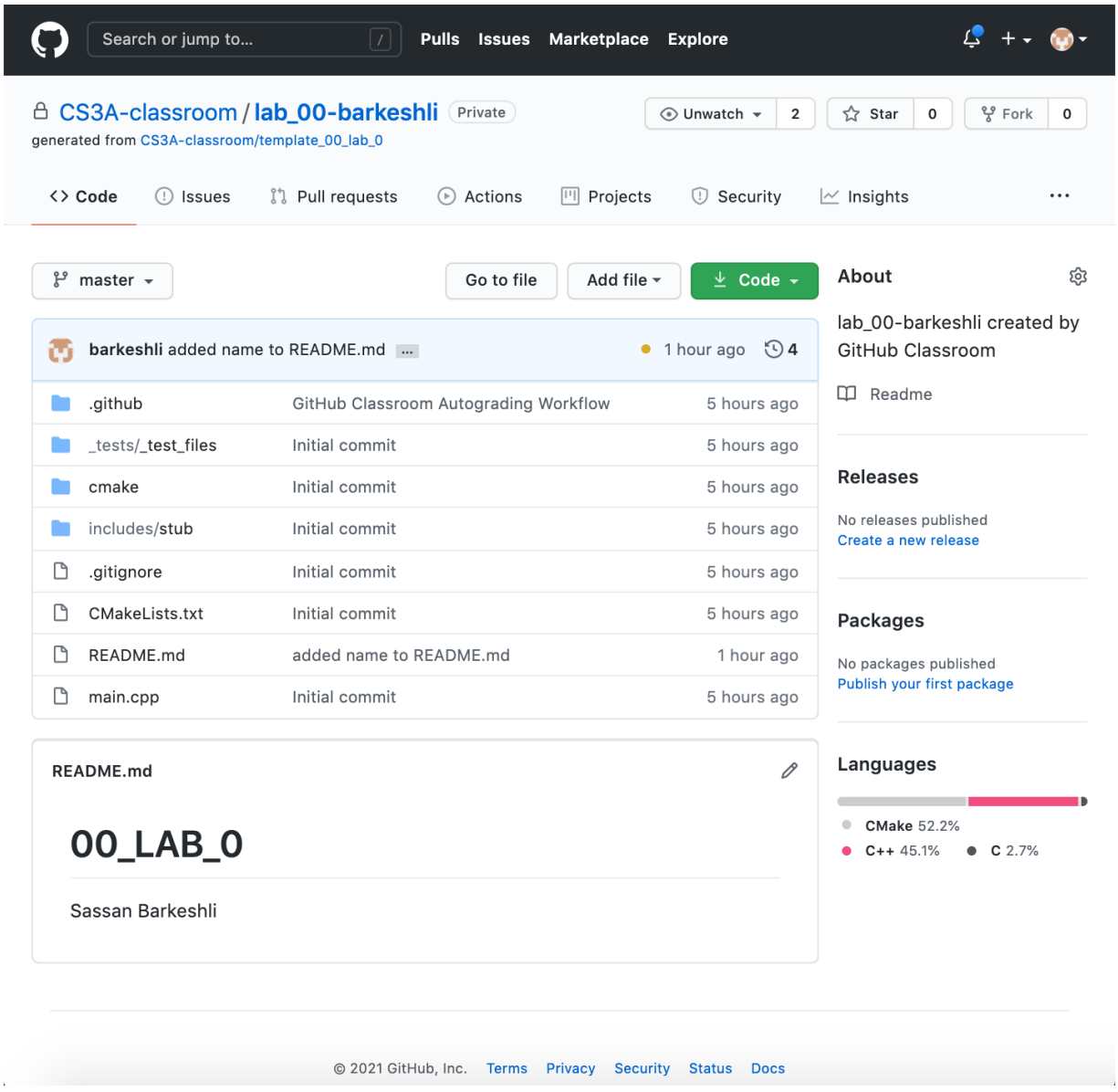
Since my branch name is `master` , I will issue the command `git push origin master` which means push to `origin` , which is my github remote name, from `master` which is my local branch name. For you, your branch namge might be `main` , so, your commmand will be `git push origin main` :

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



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:



commit and push often:

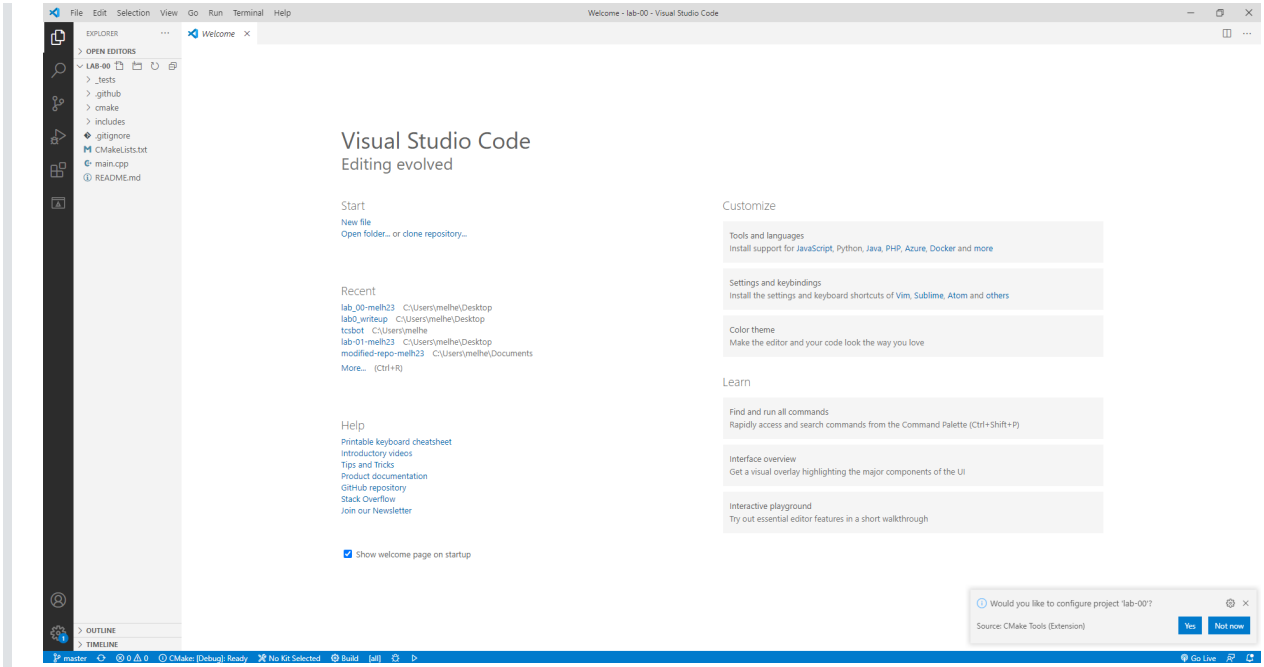
Make sure you `commit` your changes after completion of **every** task. Remeber, `commit`s are the record of your progress. You will be graded on your commits. And push your changes frequently.

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 excersise to show you how to run it.



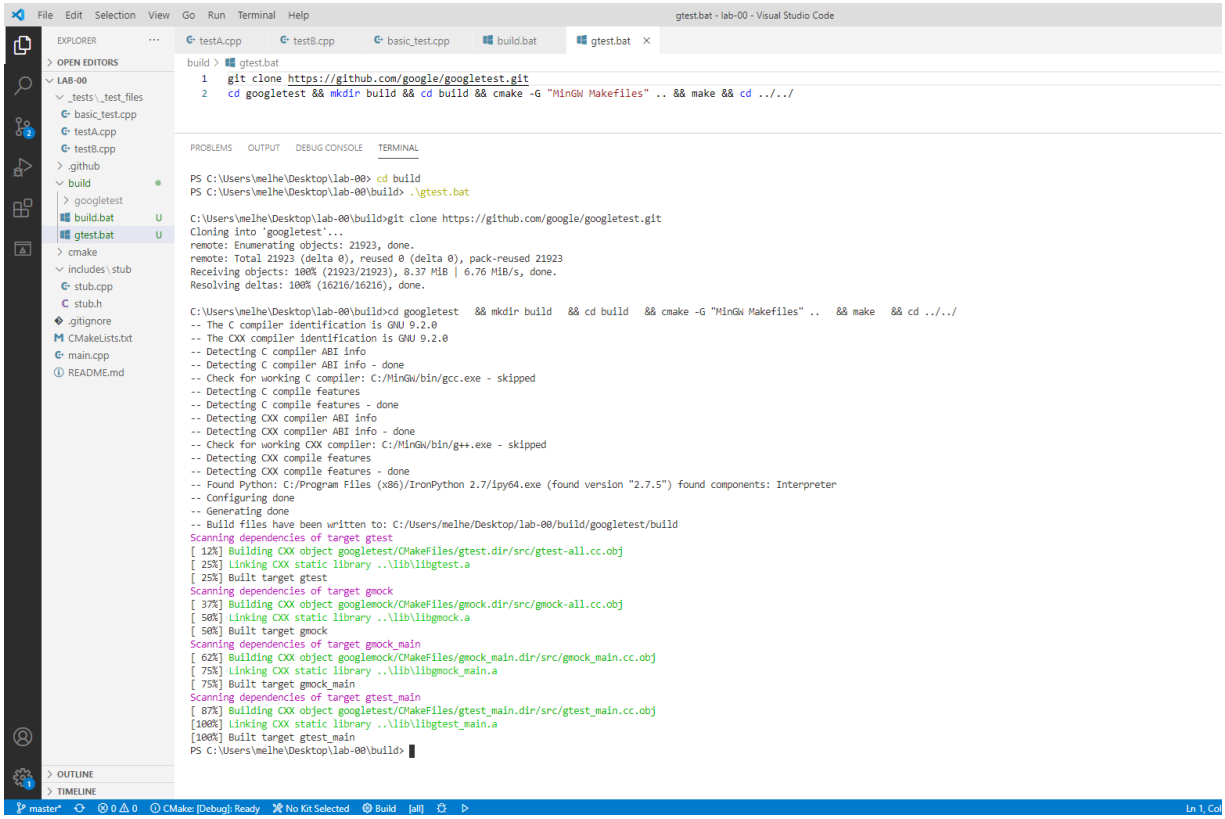
Build the googletest framework

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.

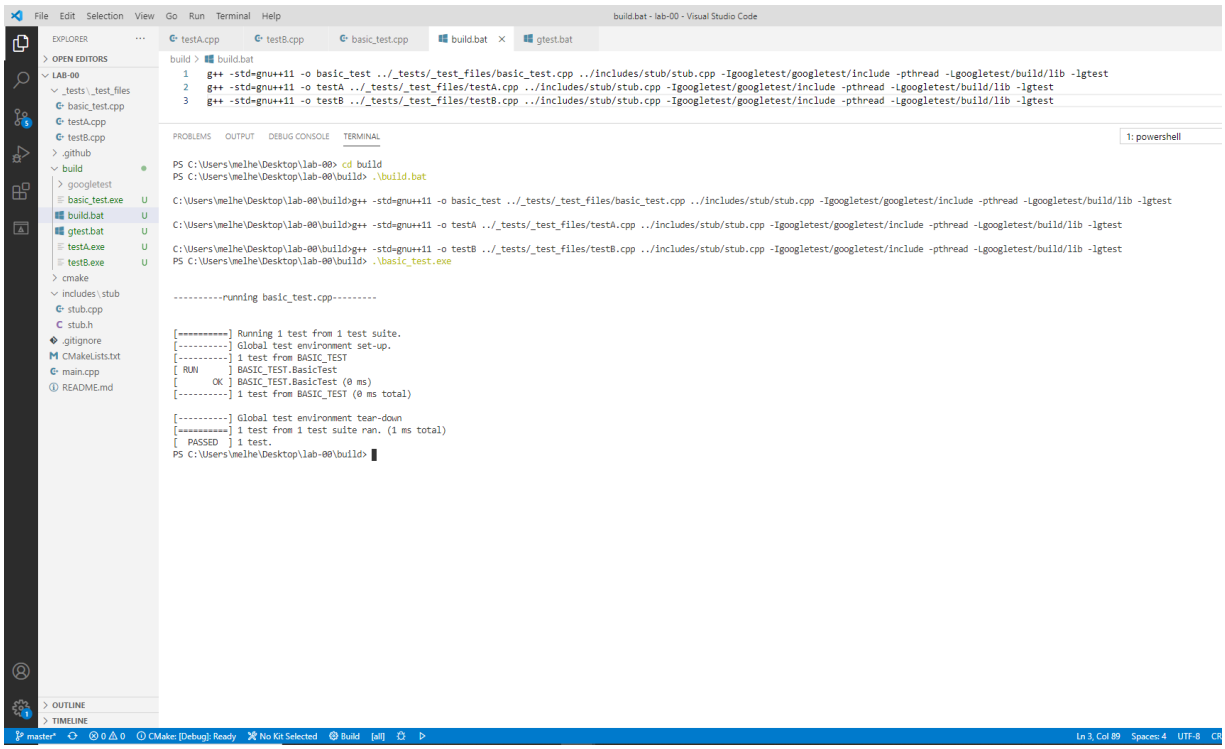
The two .bat files you will use to build and compile your project: gtest.bat and build.bat will be included with your startup code for every project.

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 auxilary files are created inside the `build/` folder. The output should look like this.



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.



Getting started with the project

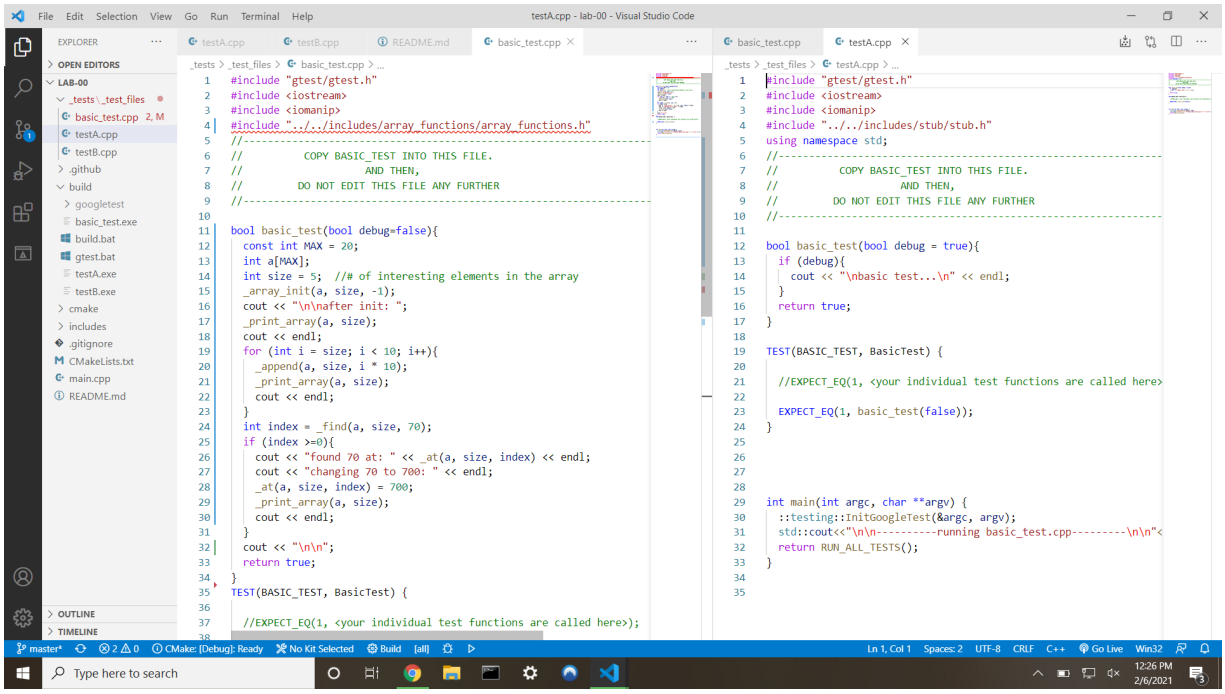
Find `basic_test.cpp`

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` demonstrtes 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.

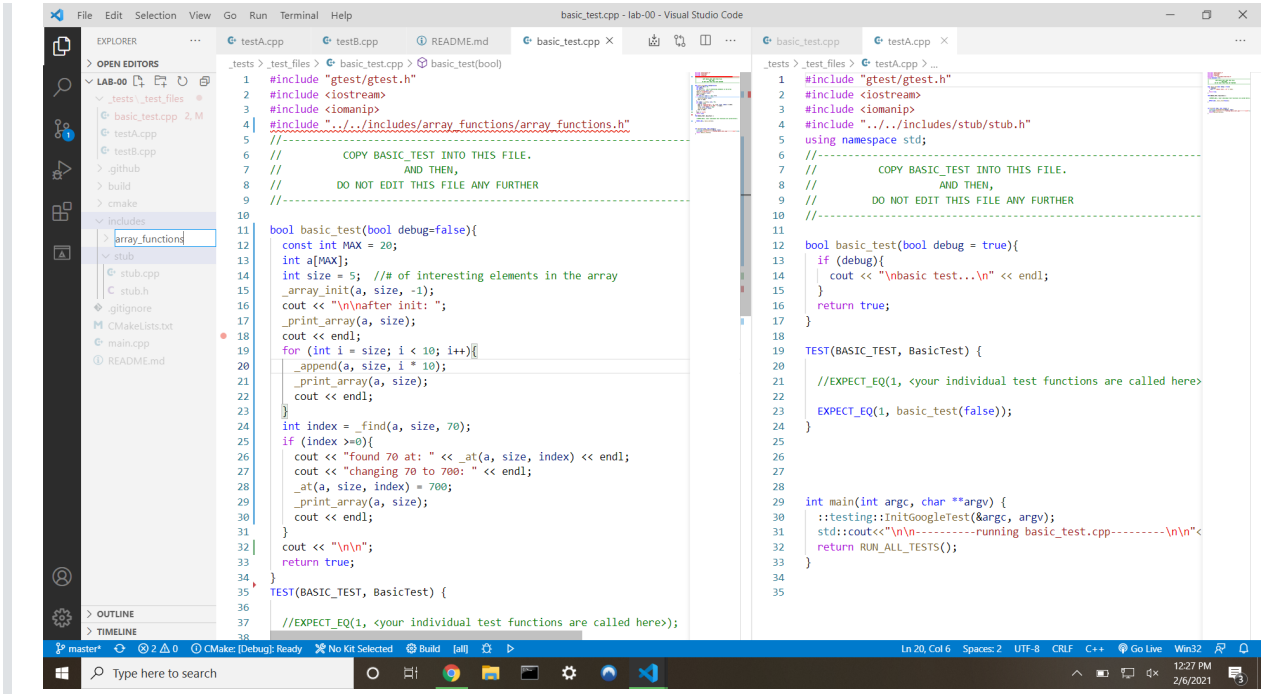
click here to download basic_test.cpp if you have not already.



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

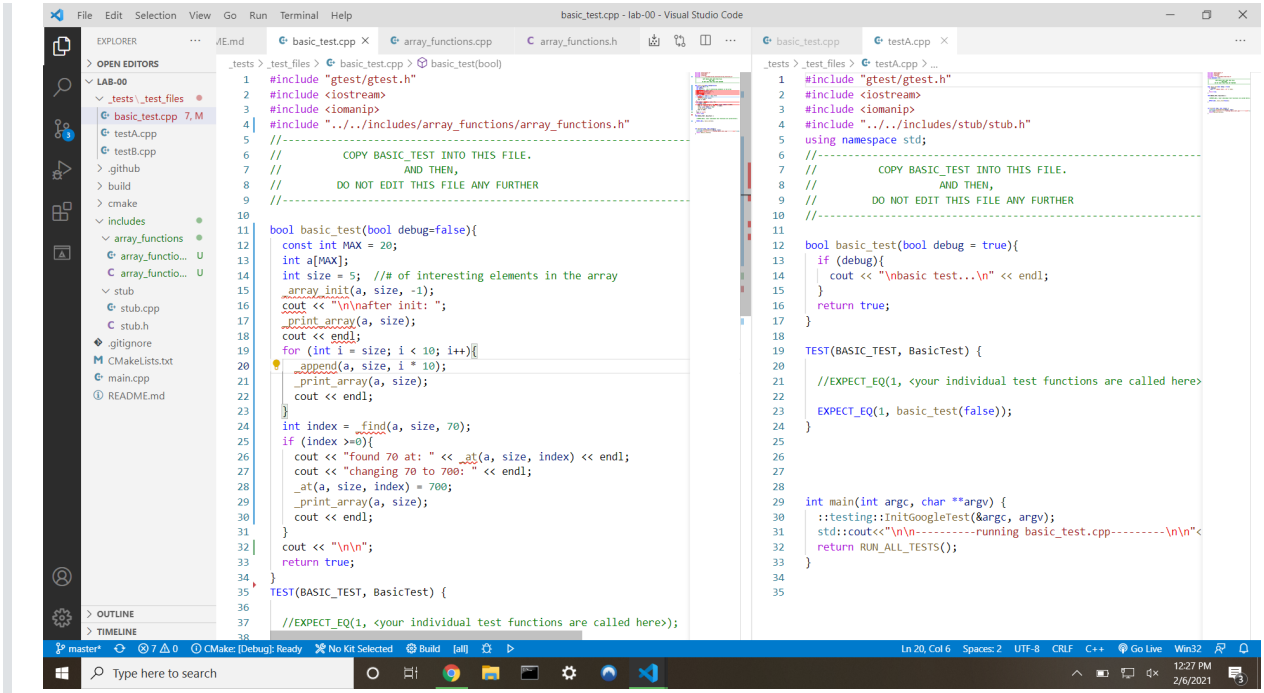
name this folder `array_functions`.

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



Add two files to this folder.

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

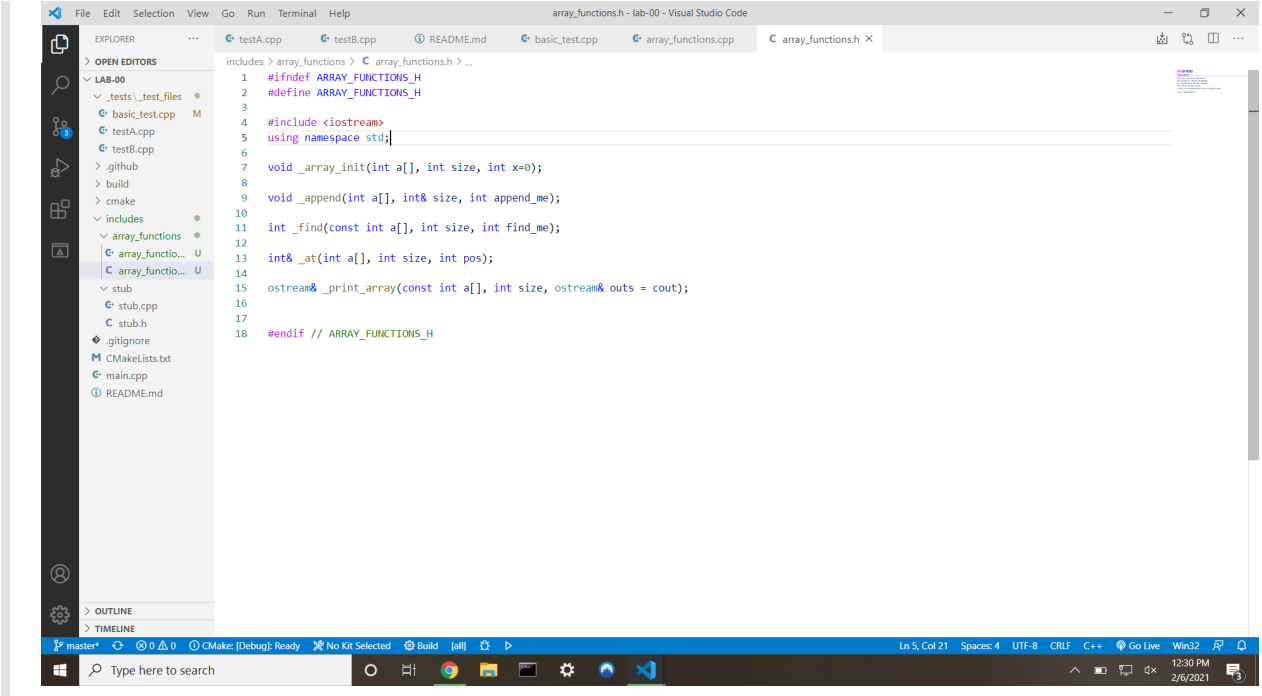


Add the function signatures.

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`



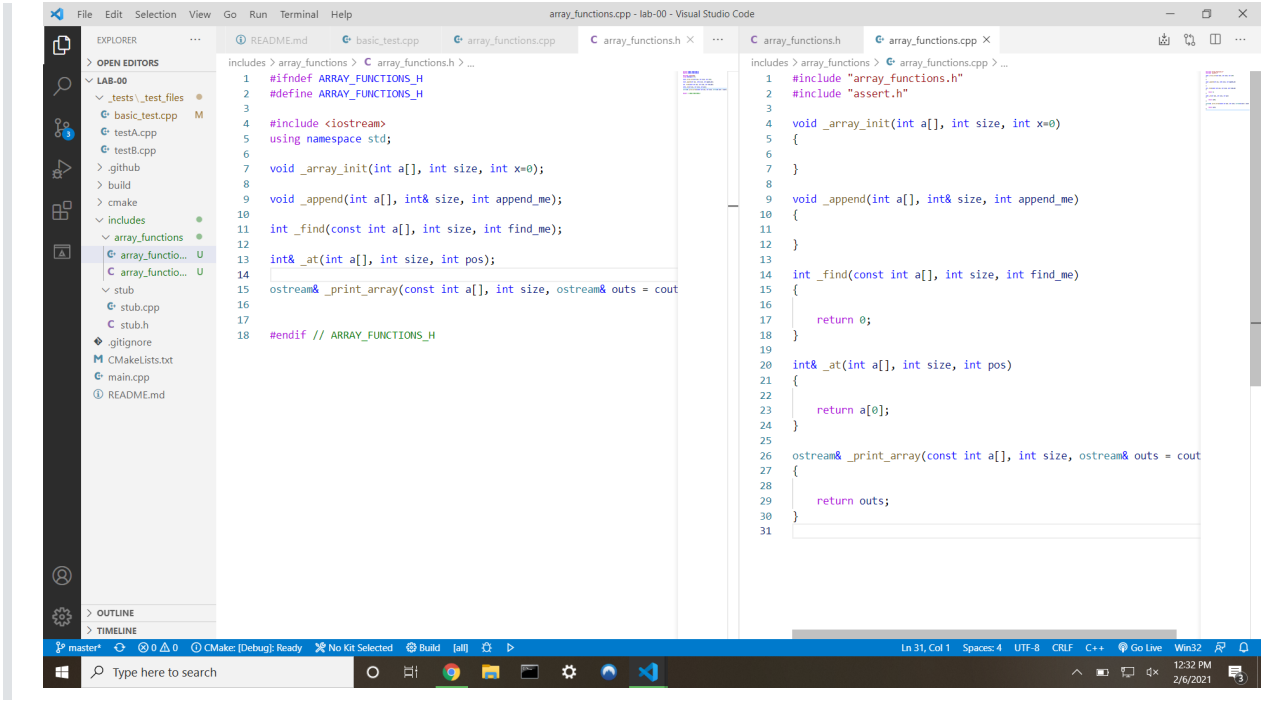
Write function *stubs*

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.



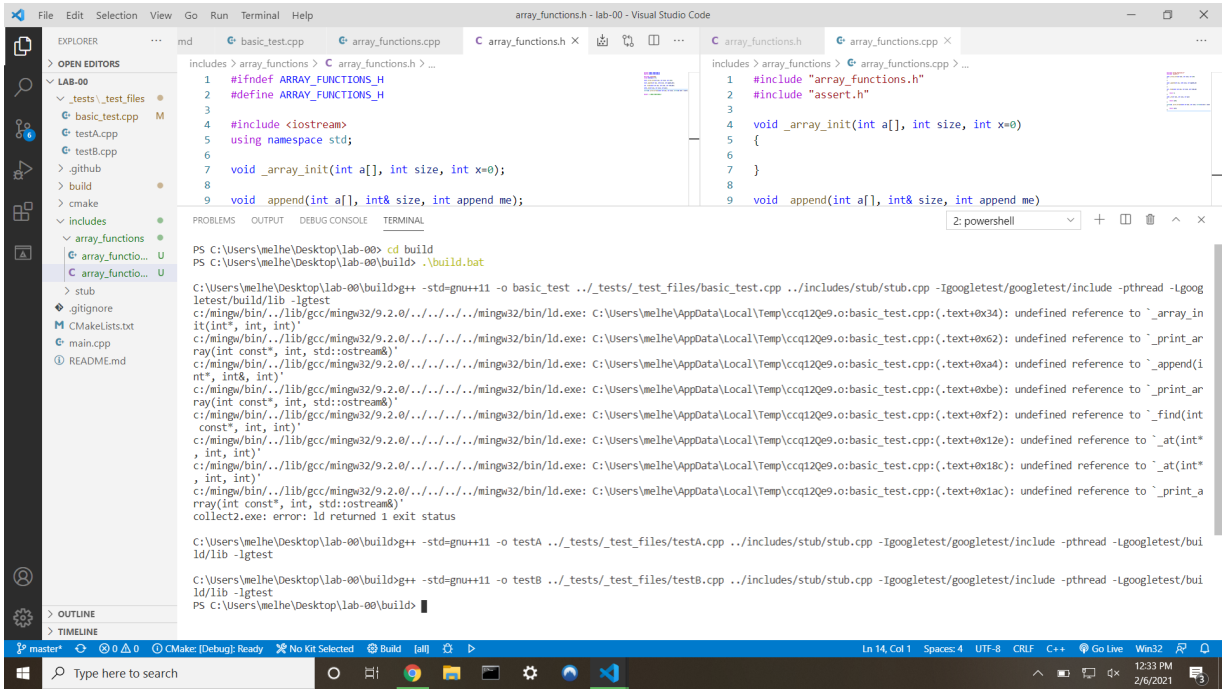
Build googletest framework

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!**



`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`.

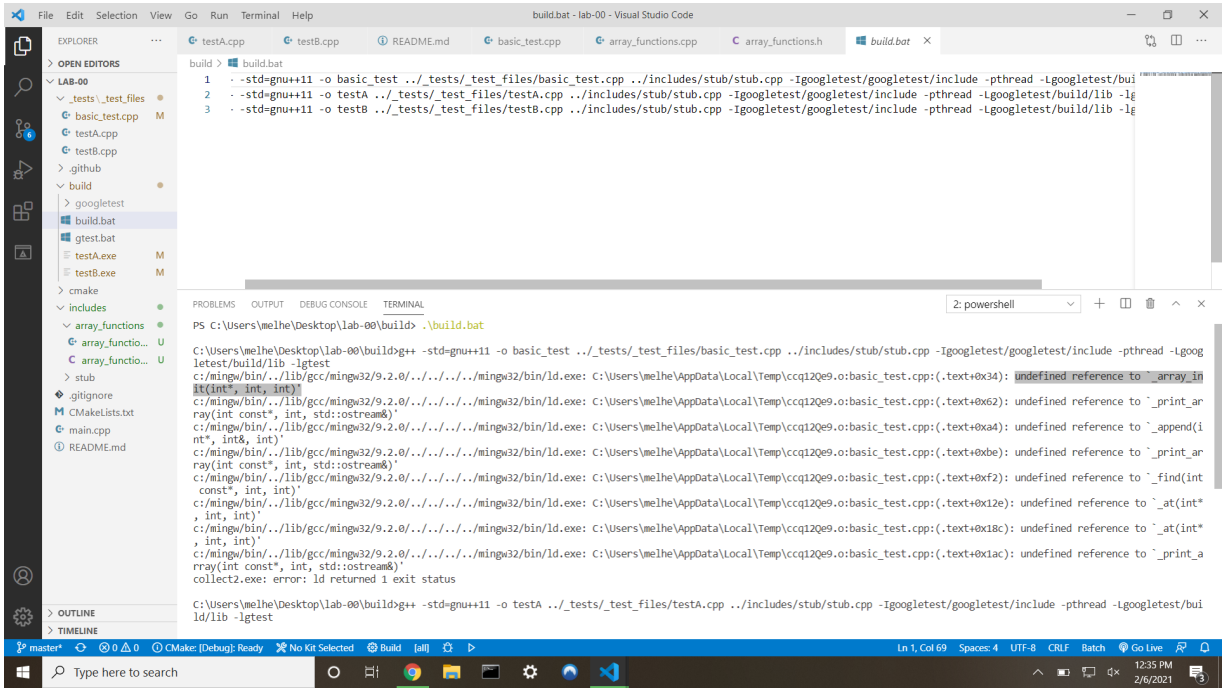
A brief explanation of the line of code in buld.bat follows:

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

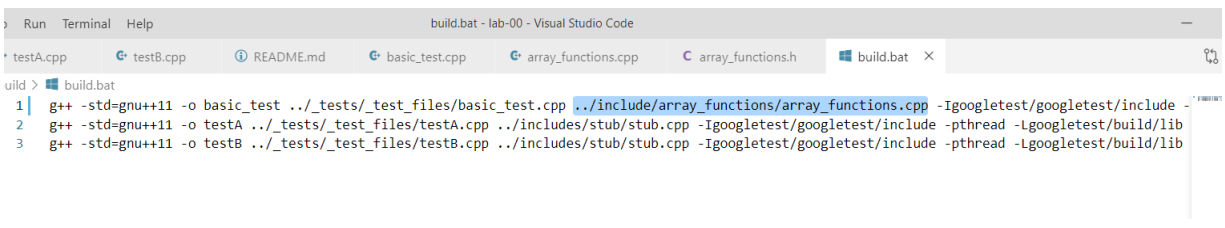
`basic_test` in this command is the name of the executable that will be created.

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.



Editing the batch file

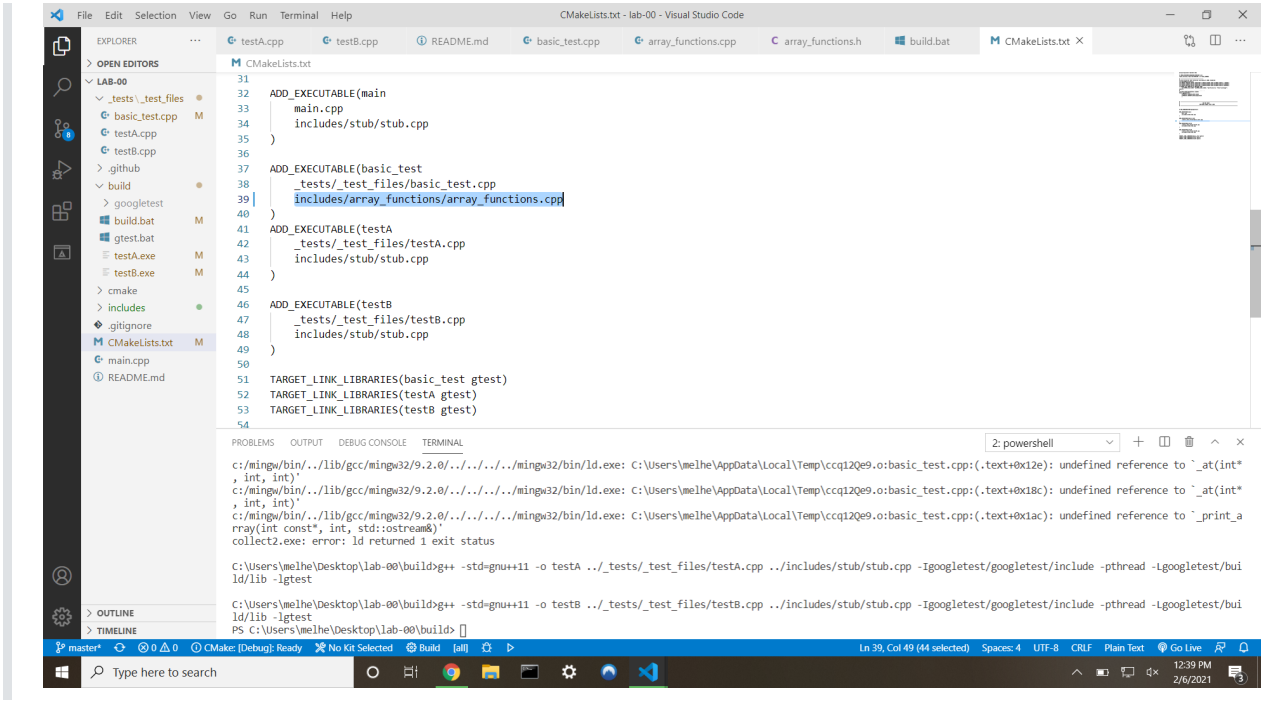


back at the `CMakeLists.txt` :

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...

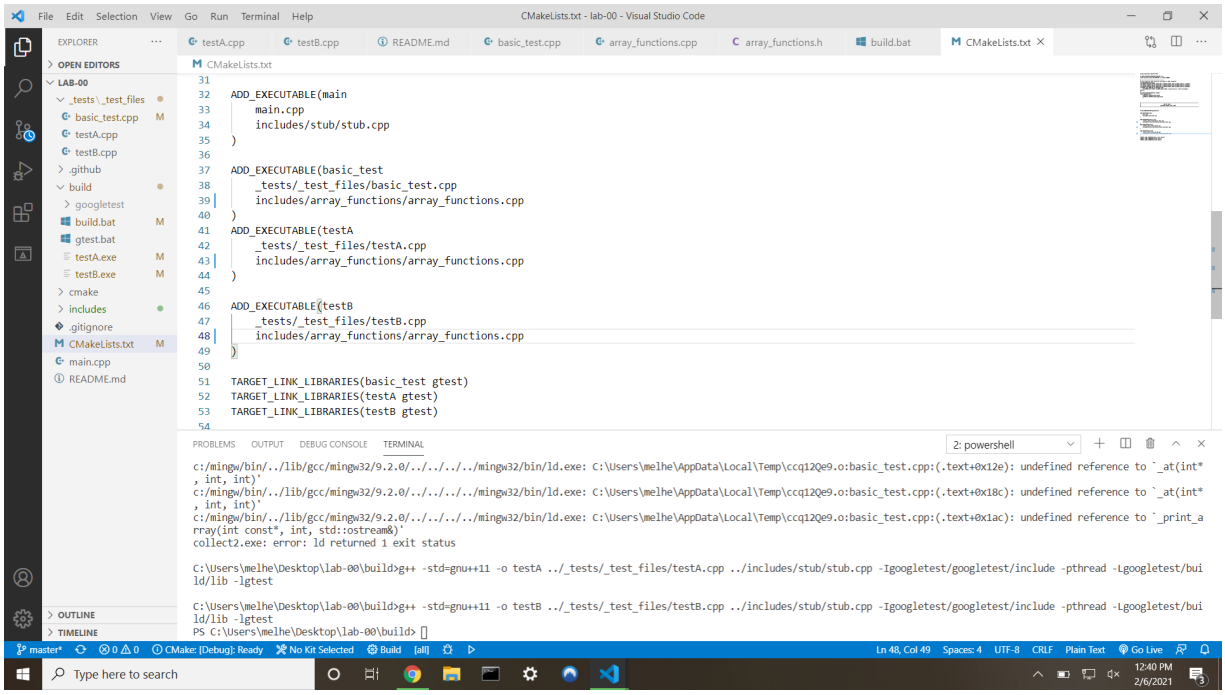


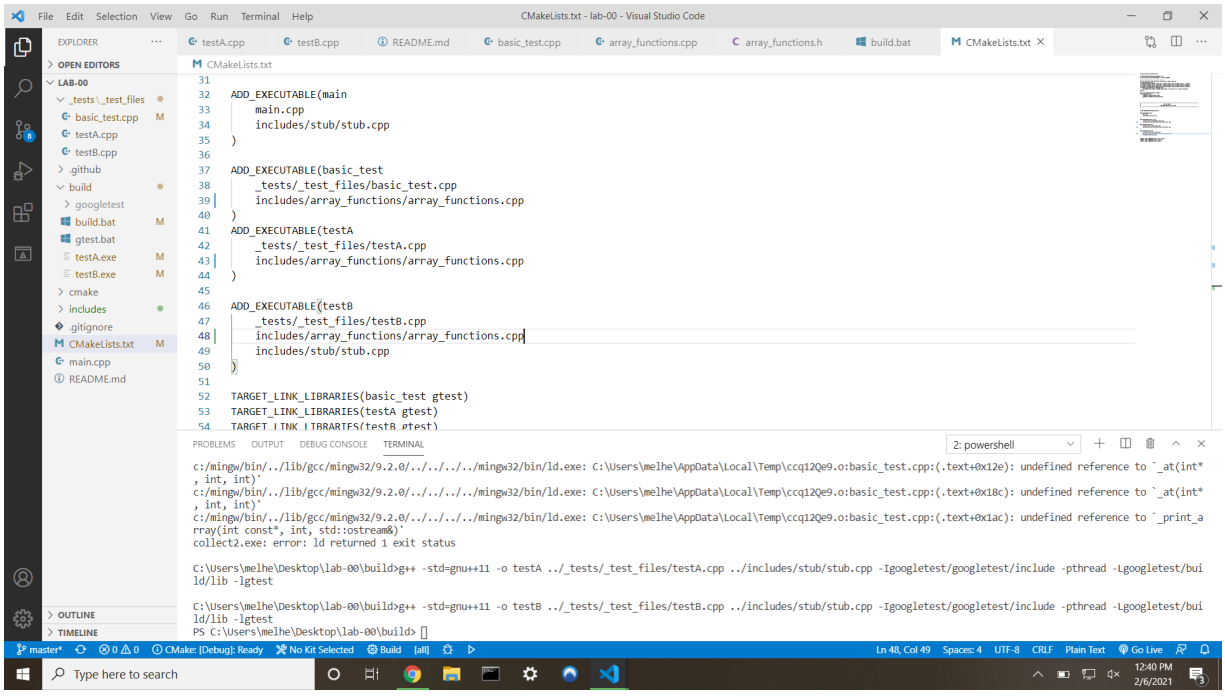
Add `array_functions.cpp` to the `ADD_EXECUTABLE(basic_test... )`

Do not use commas to separate the files.

Do NOT include `.h` files.

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



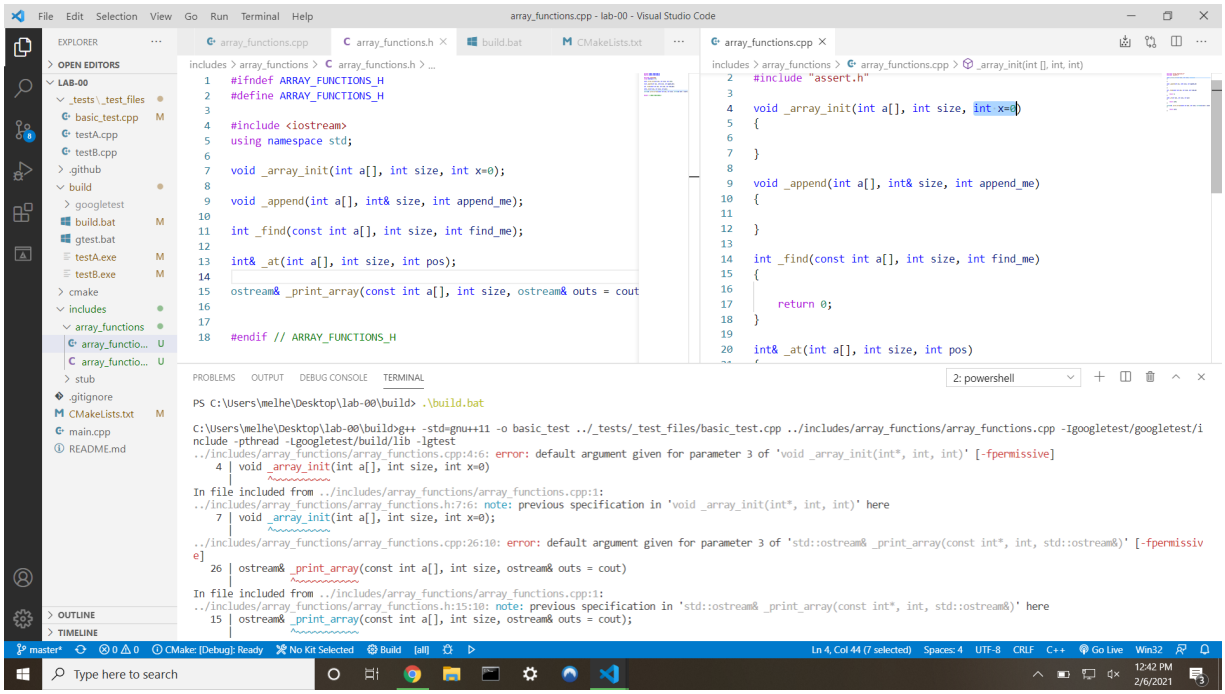


`.\build.bat` again:

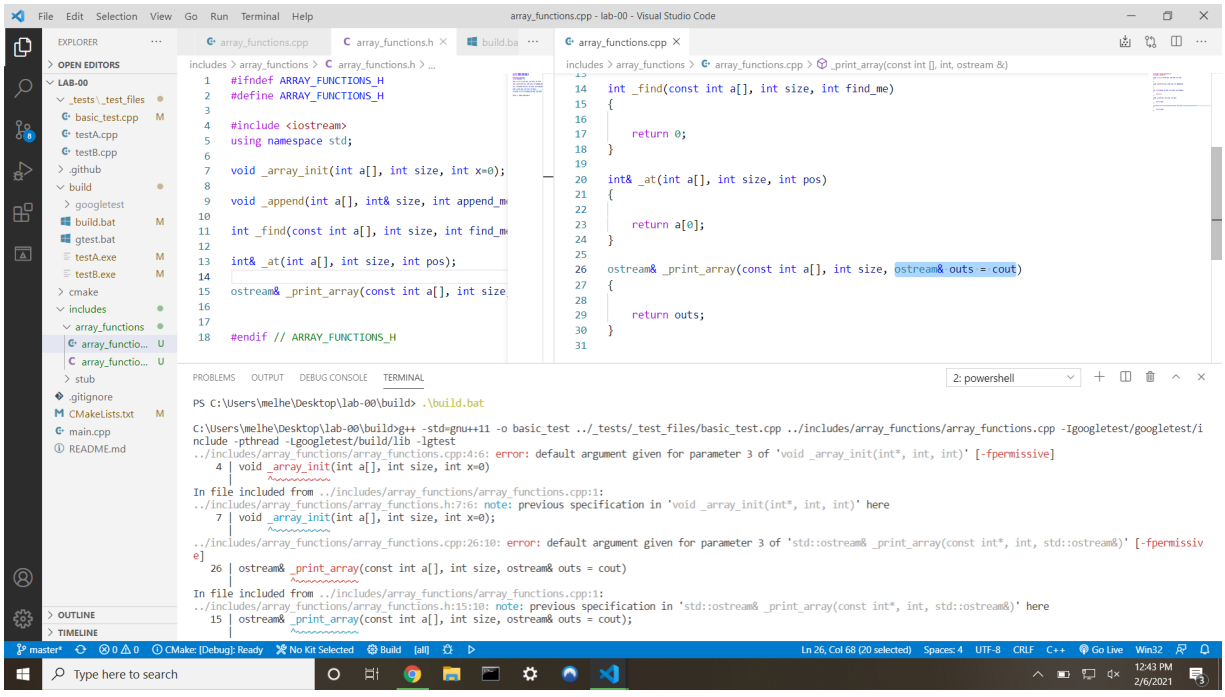
Let's run `.\build` again and pray that...

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

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

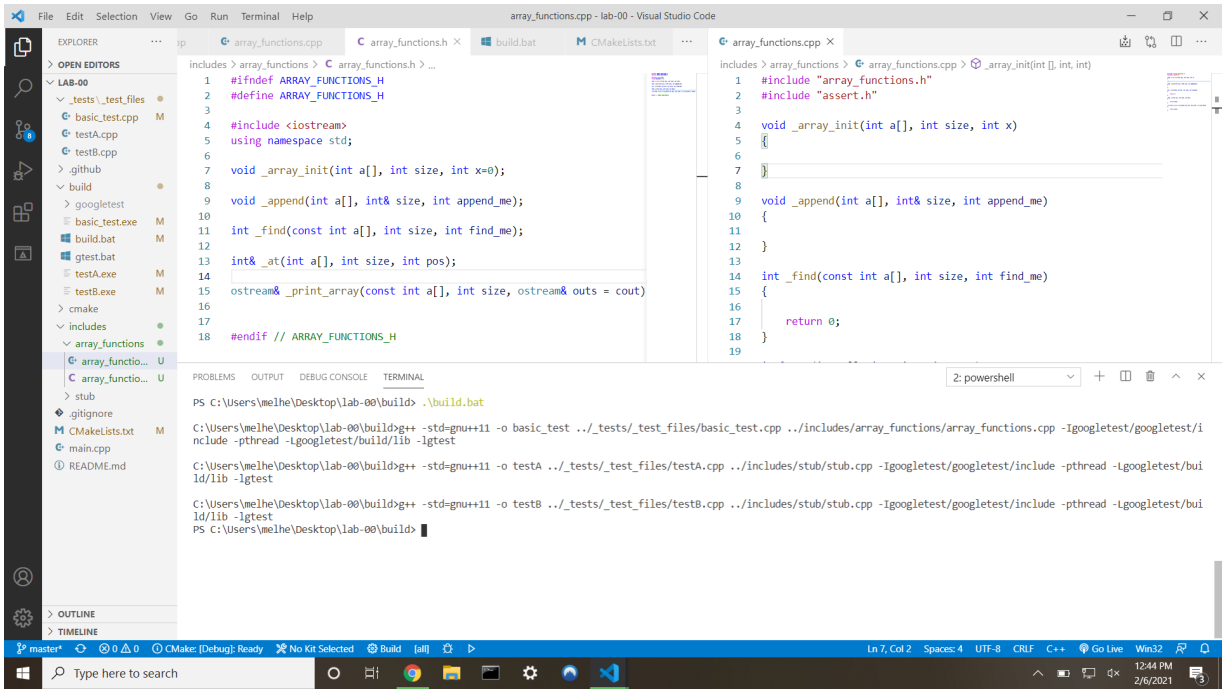


Fix the `_print_array` function...



by removing the default value `= cout`

Same with `_array_init`:

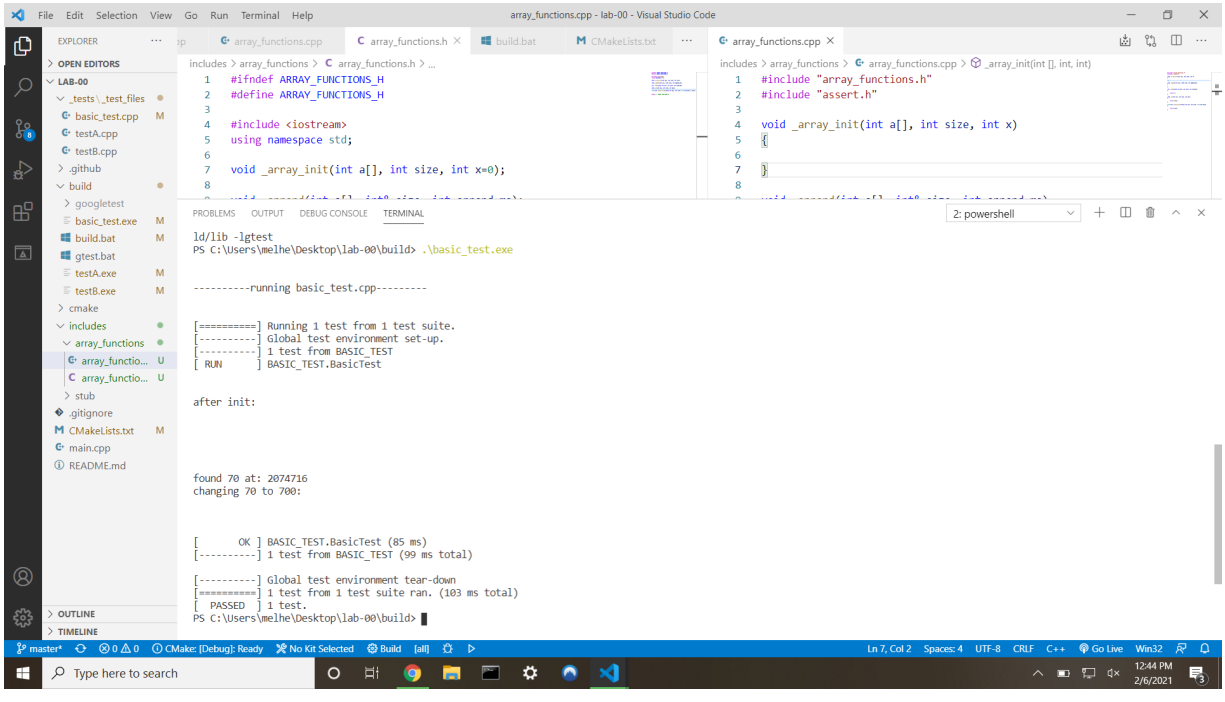


`.\build.bat` one more time:

and this time it will run successfully.

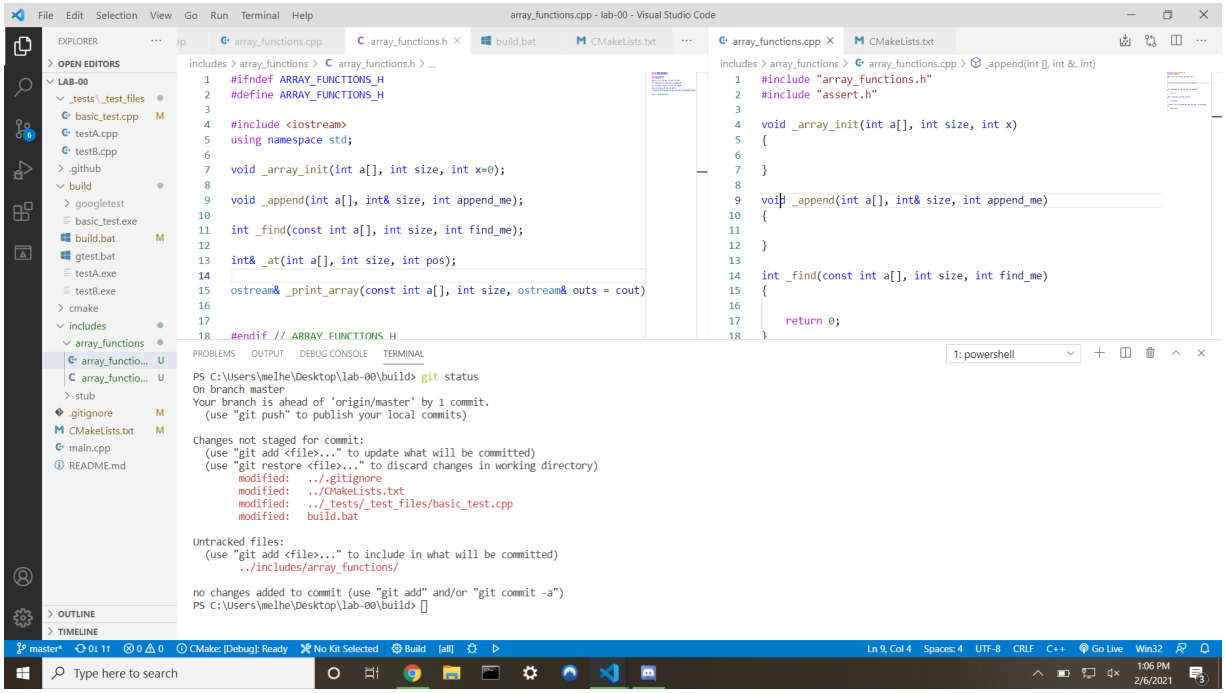
This is a huge step. We now have a working project eventhough our functions are basically empty.

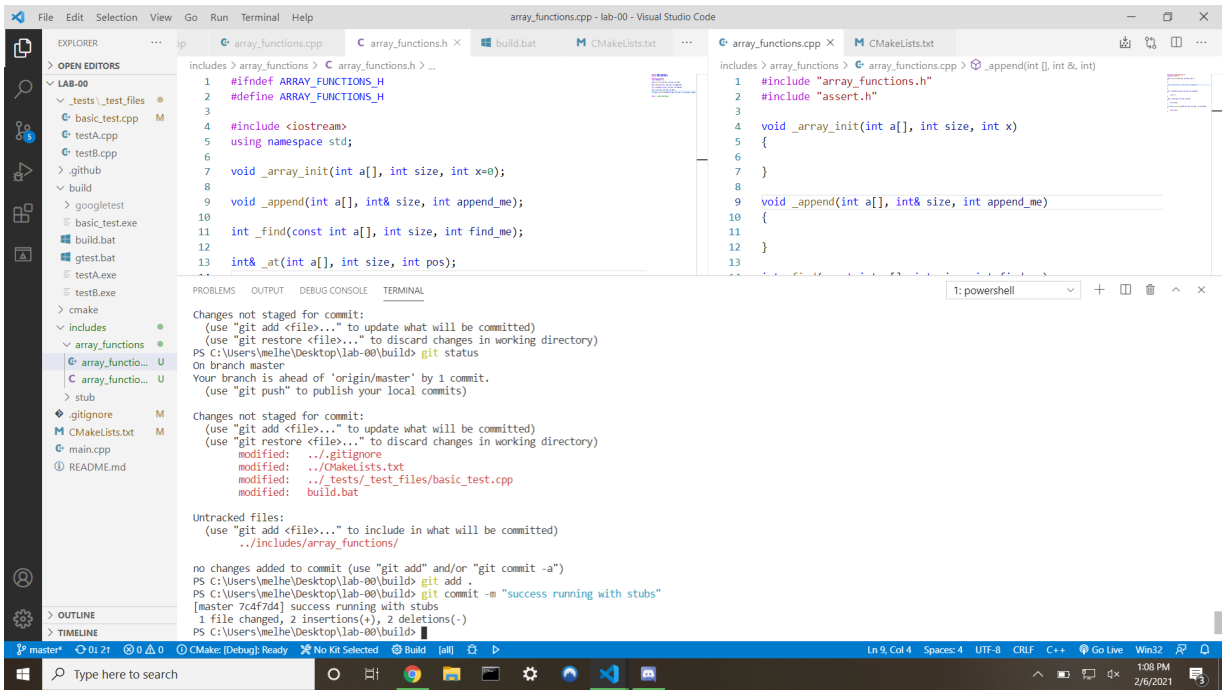
You can even run `basic_test.exe`. Of course this will not run satisfactorily. You will get mostly garbage. -afterall, we are running on stubs!- but it **does** run!!



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.





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 otgherwise.

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:

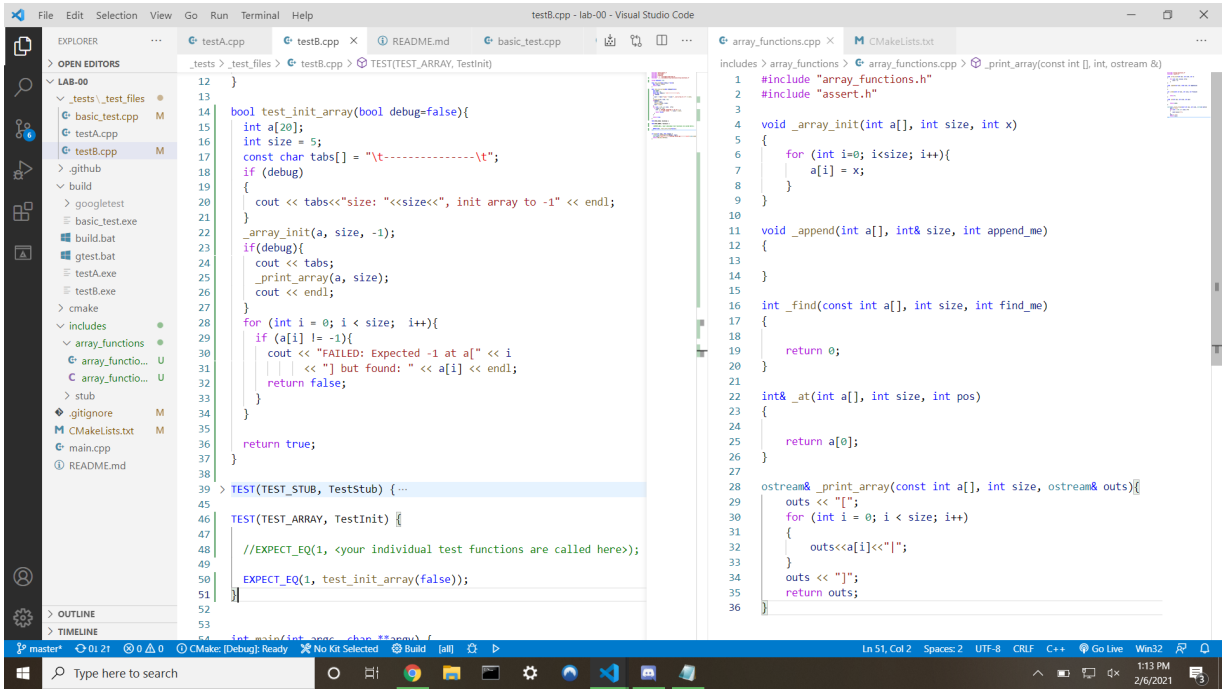
The `TEST` function is part of the googletest testing framework. To simplify our work, we always use the same format for the `TEST` function: Declare a `bool success` and assign it to the return value of the test function.

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.

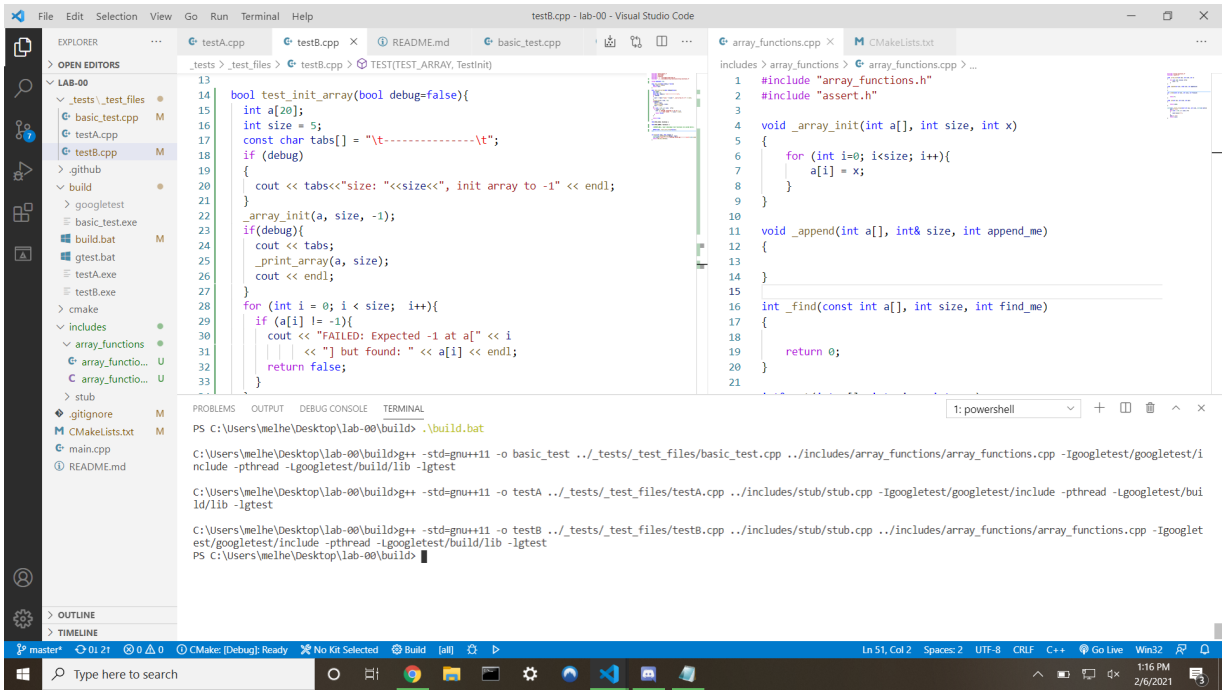


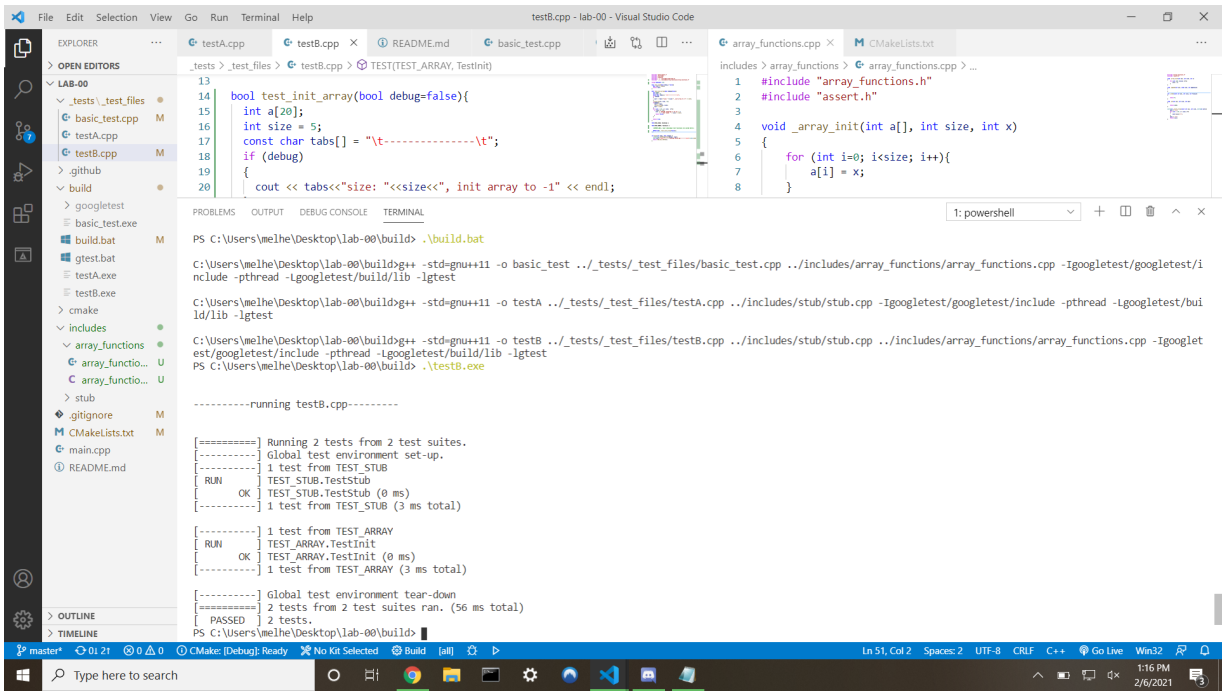
build and RUN!!

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`.



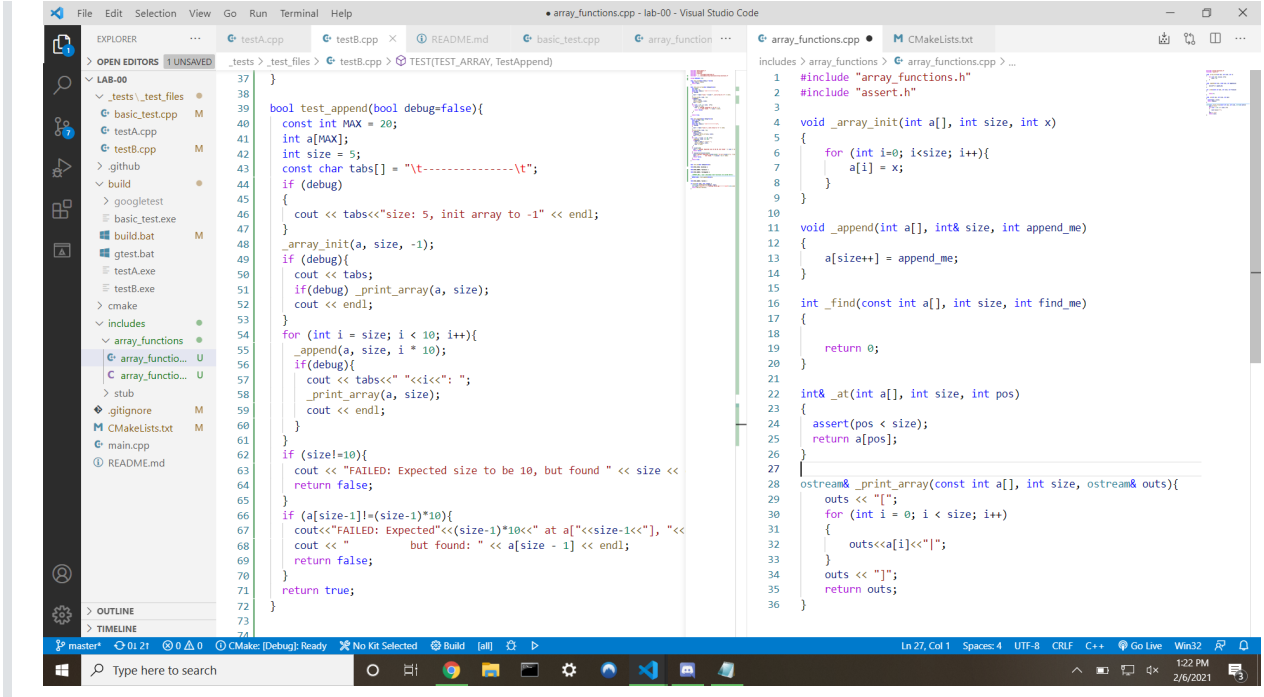


Implement `_append` and `_at`:

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

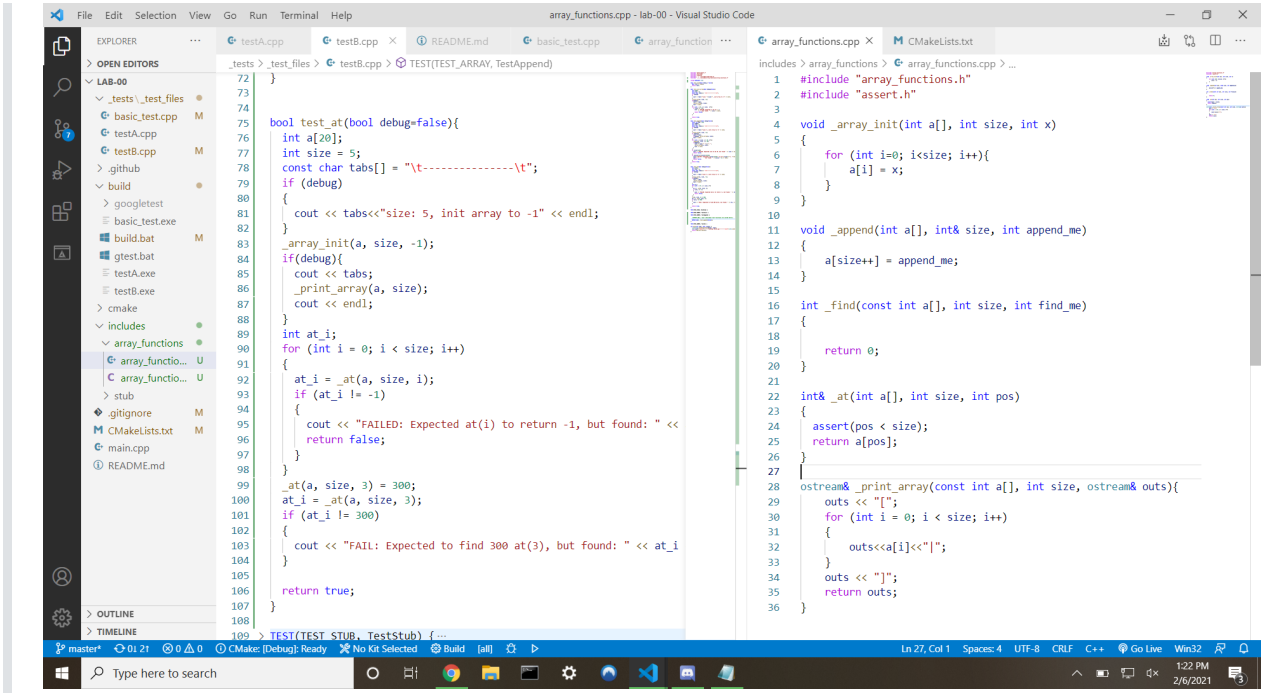
You will *borrow* my code for this particular lab, but make sure you comment the code very well.

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.



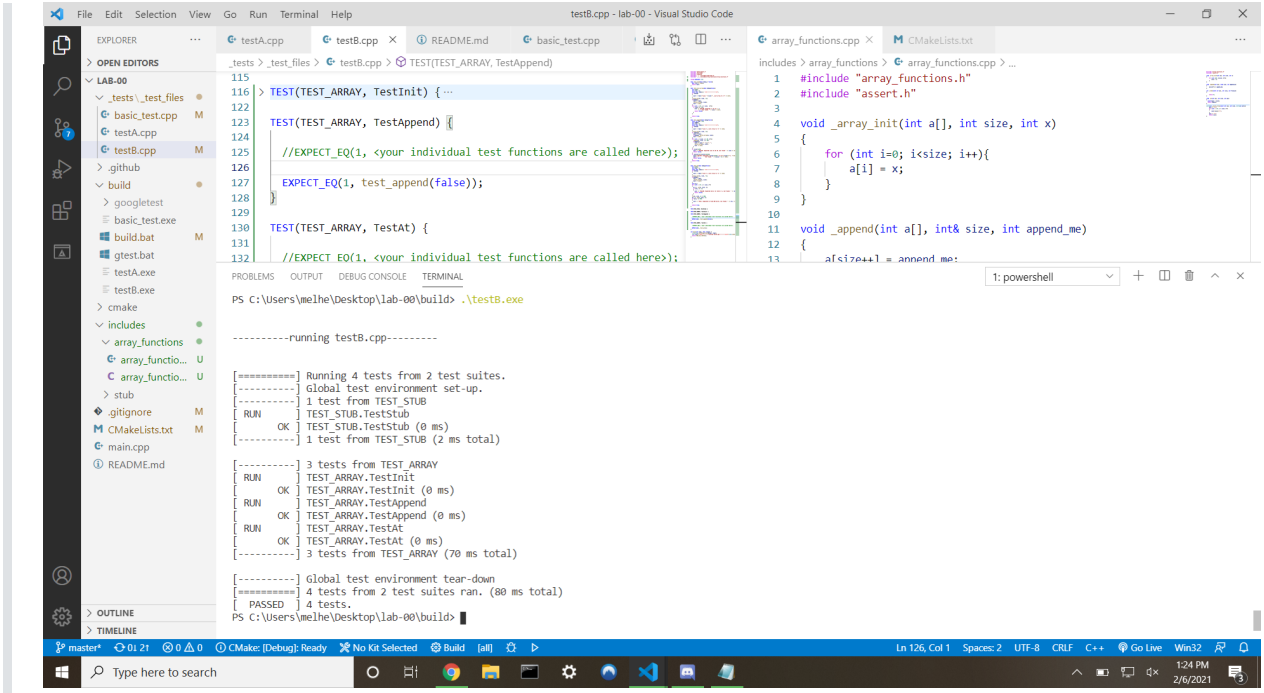
Obviously, this is done in the `testB.cpp` file. Again, do not forget to add the `TEST( )` function for `test_at()`

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



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

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

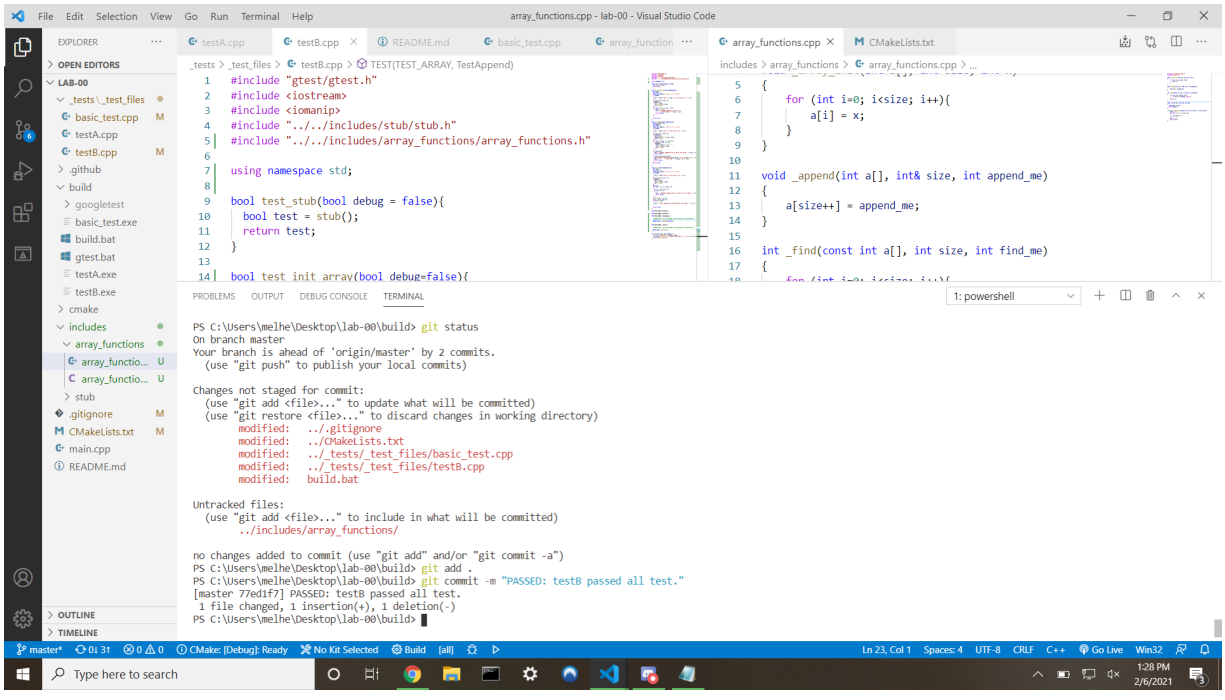


Completing the project

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.

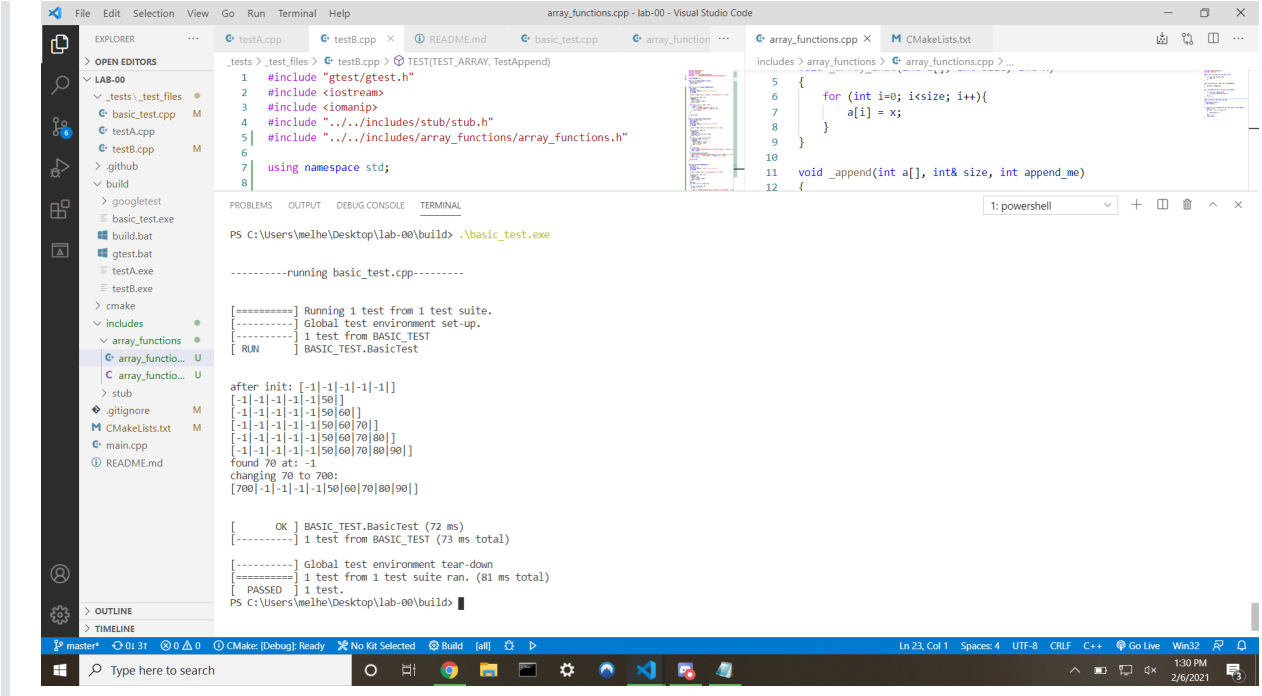
Once you have successfully run `testB` with `_find`, you `git add` and `commit` your changes. Once again, do not forget to update your CMakeLists.txt file for the autograder.



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.

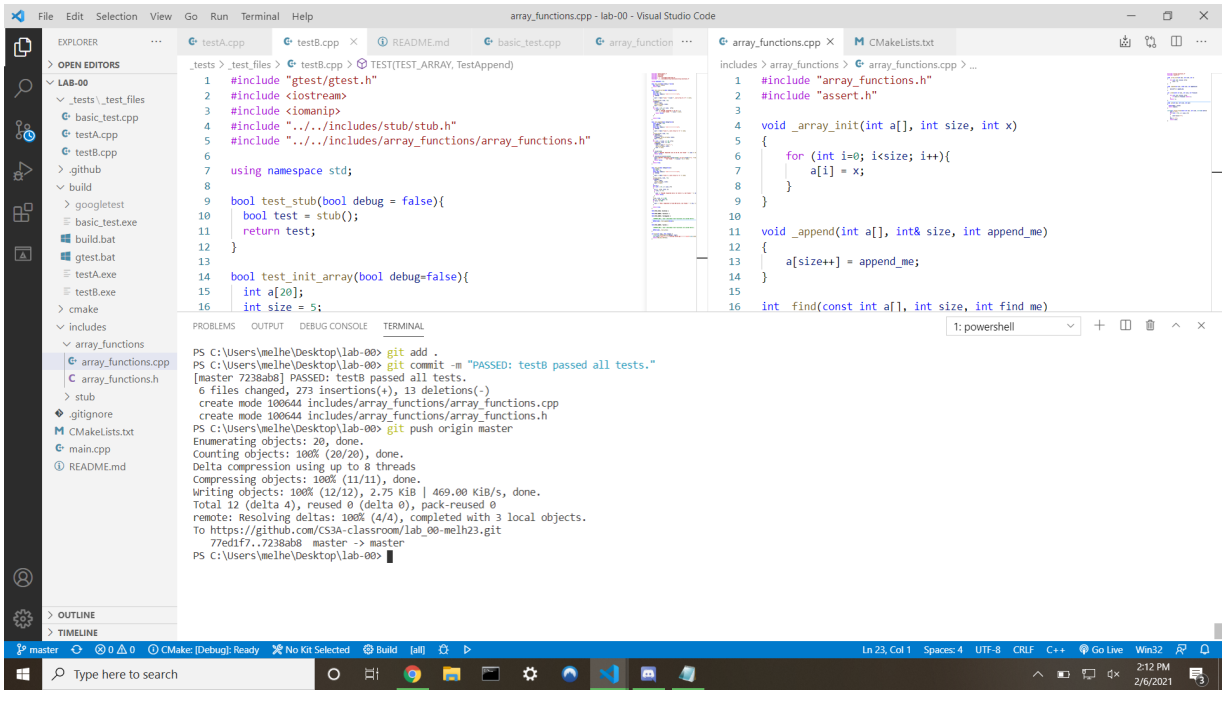
I cannot overemphasize how important it is for this test to be able to compile and run **without** your editing it in any way. If your project cannot compile and run `basic_test`, the grader will not be able to run your project.



`git add` and the final `git commit`

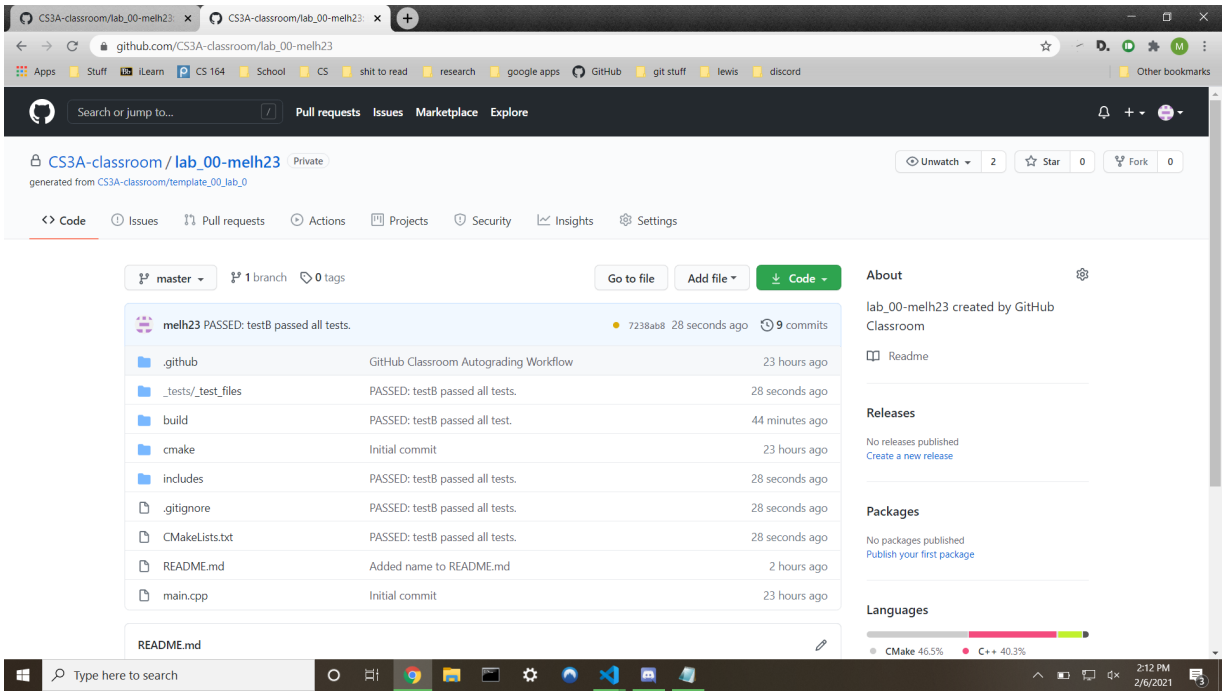
Let's go back to the root directory by typing `cd ..` - remember that `..` means parent directory. `cd ..` means change directory to the parent.

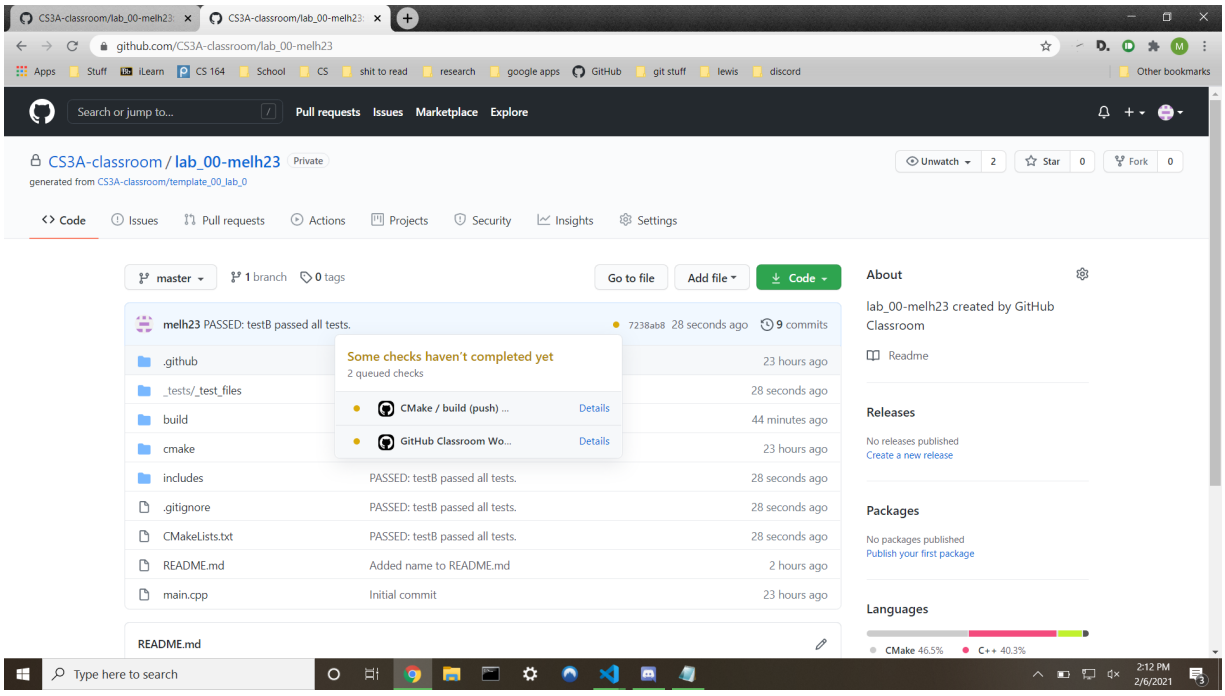
My commit message will let me know what stage of the development I am in. I have just PASSED both the `basic_test` and `testB`



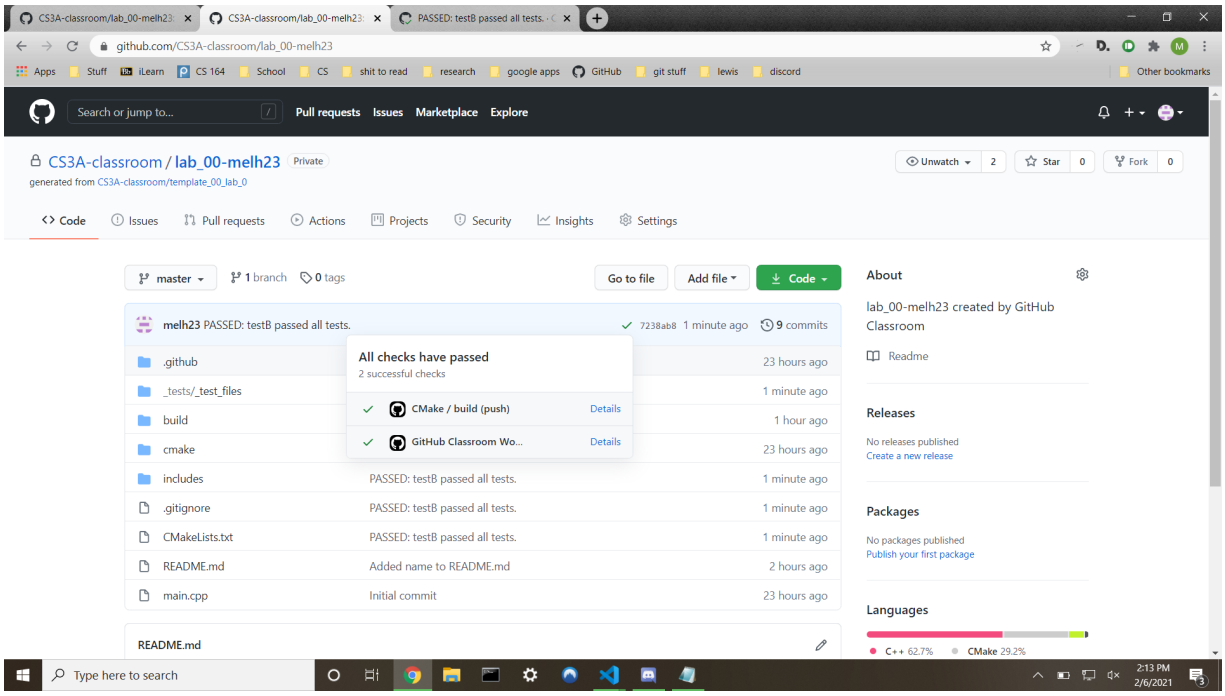
Autograder Status

You can keep track of the grading status of your project on the assigmment 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 neans that all your tests have passed. A red x means that at least one test failed.

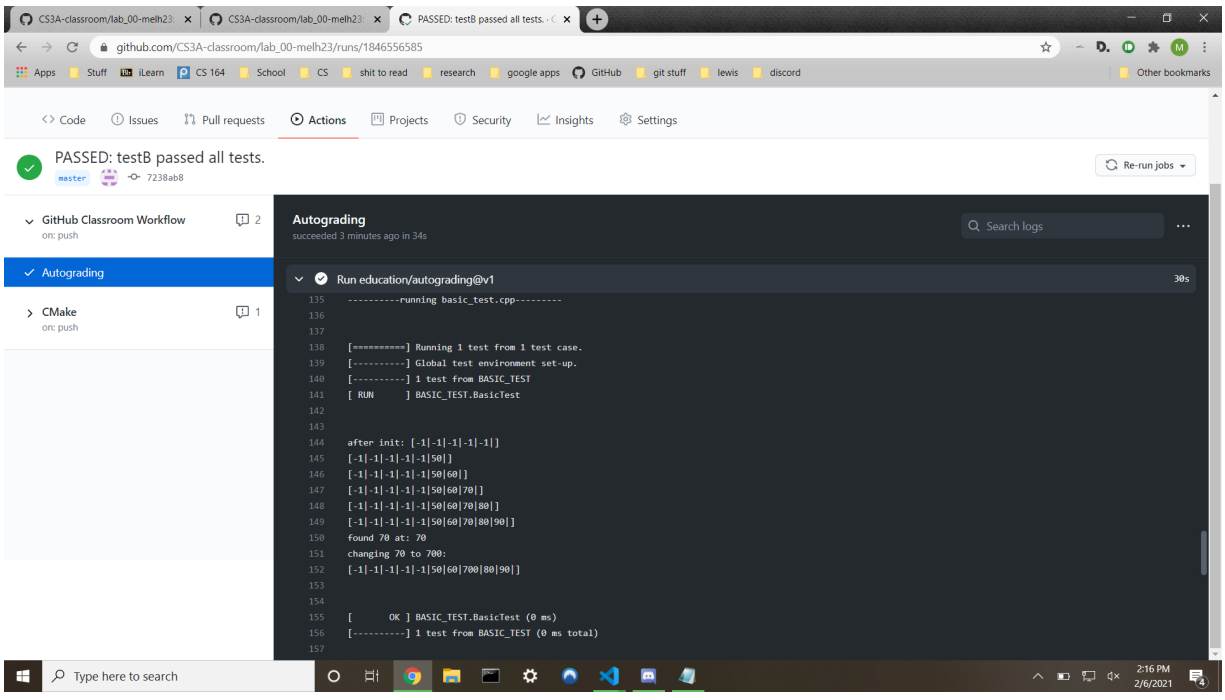


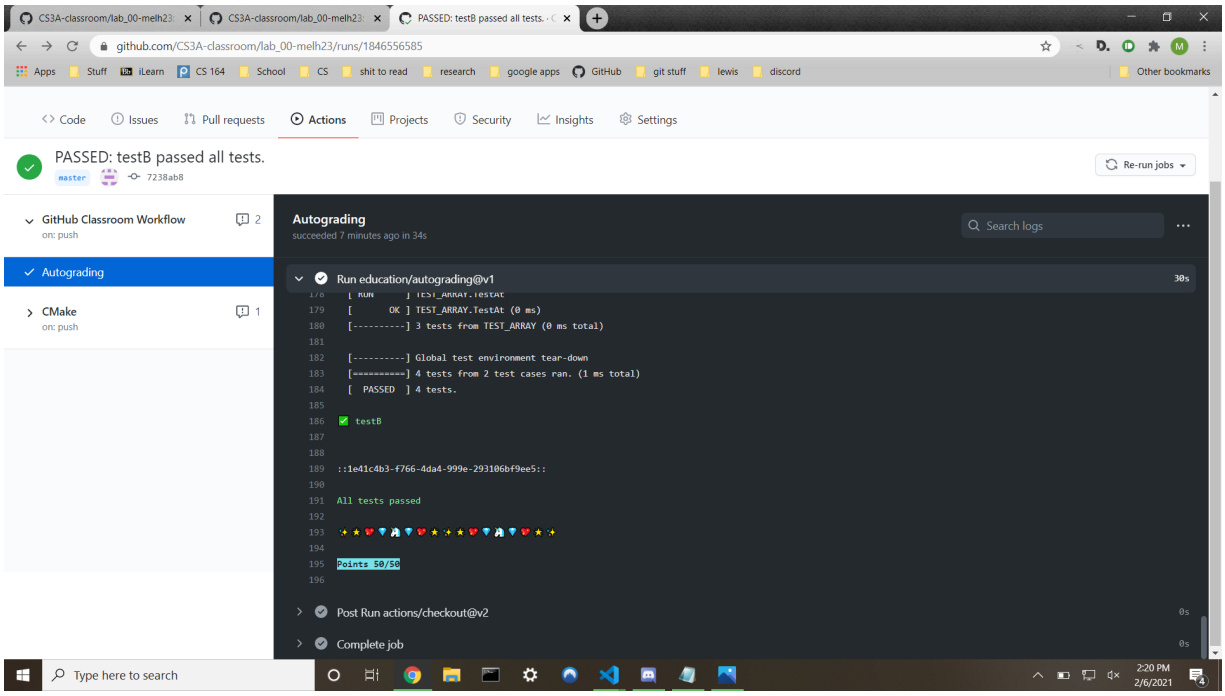


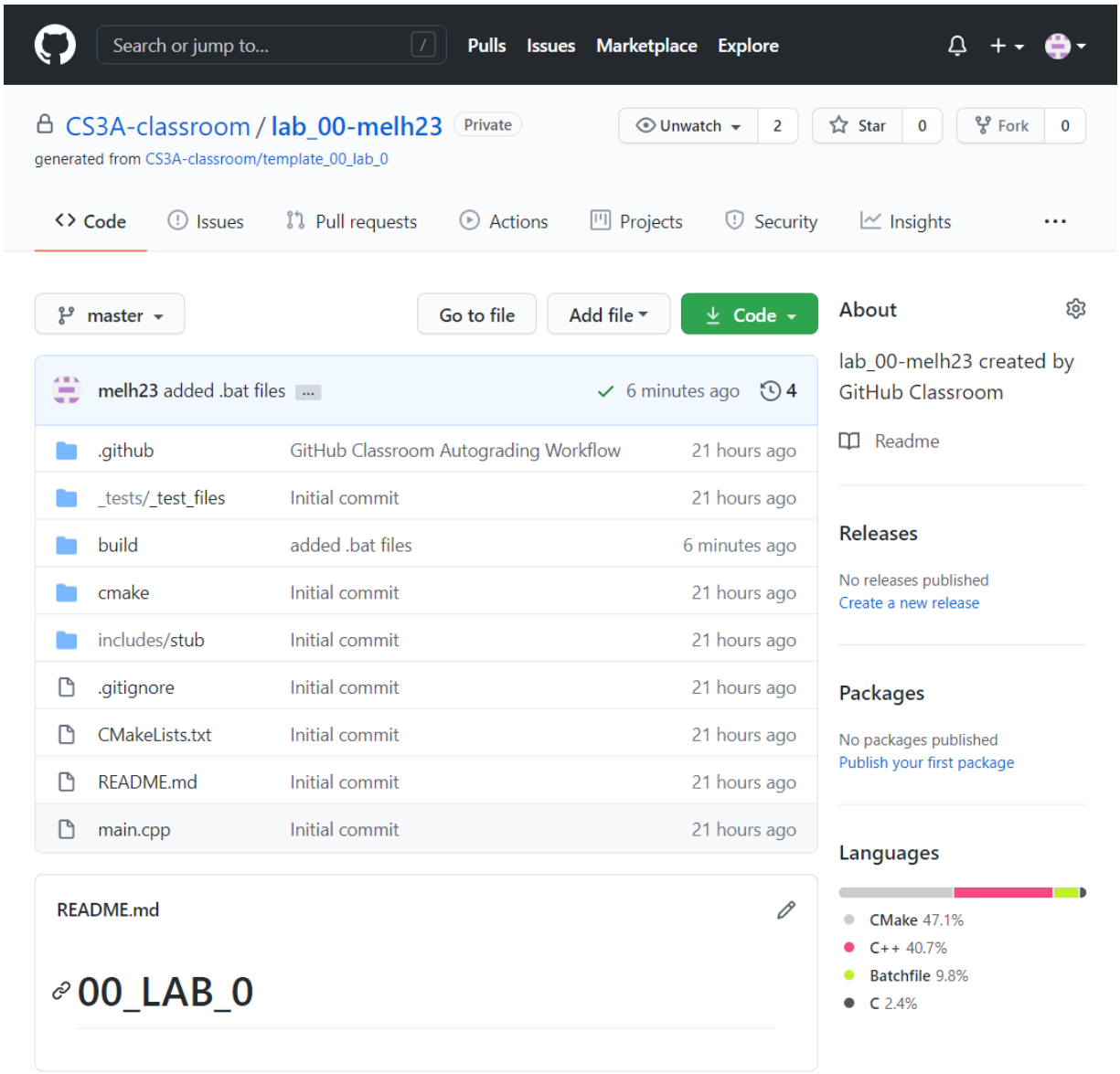
To see a more in depth output of the autograder test runs, click on the Details link:

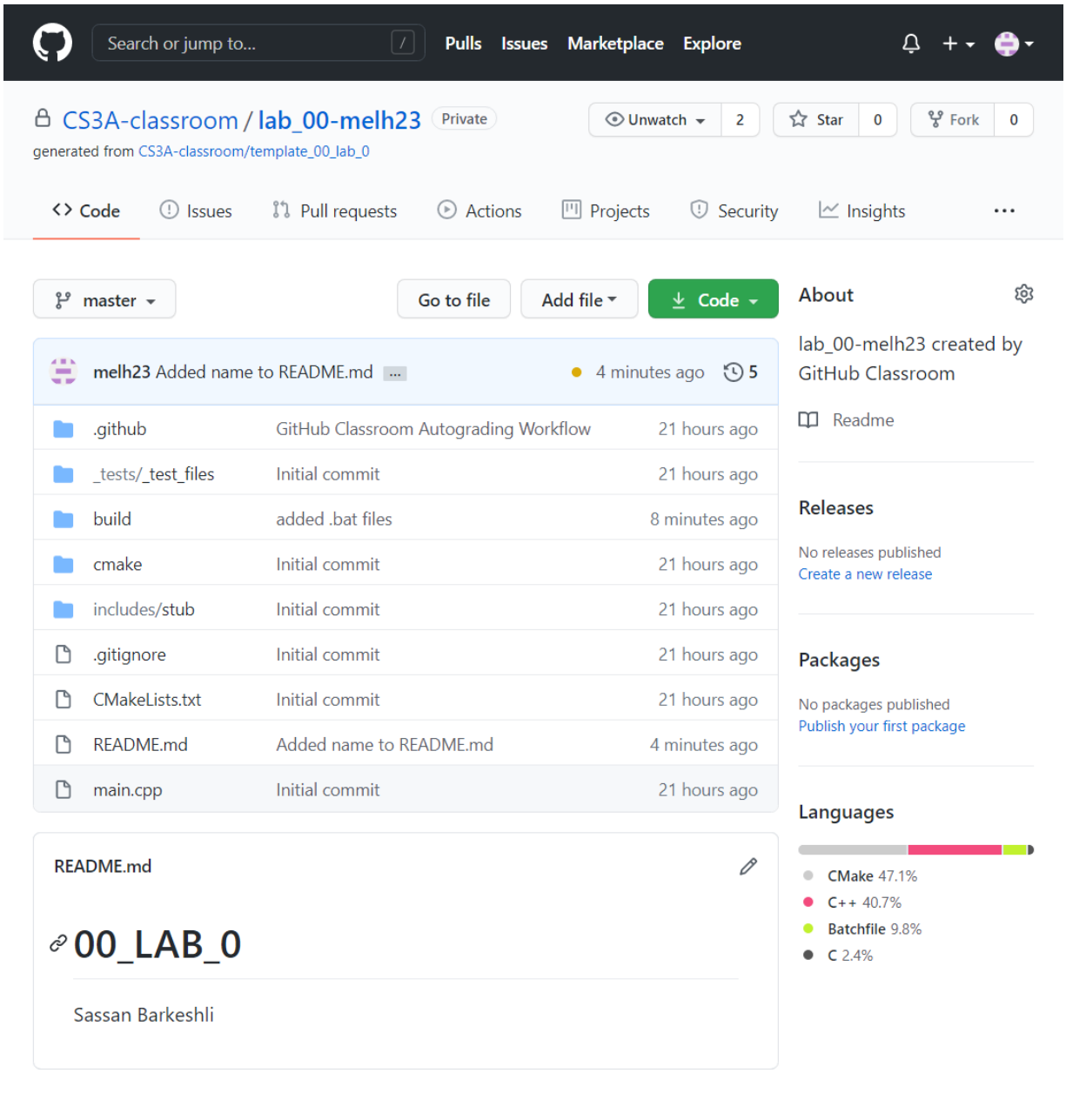


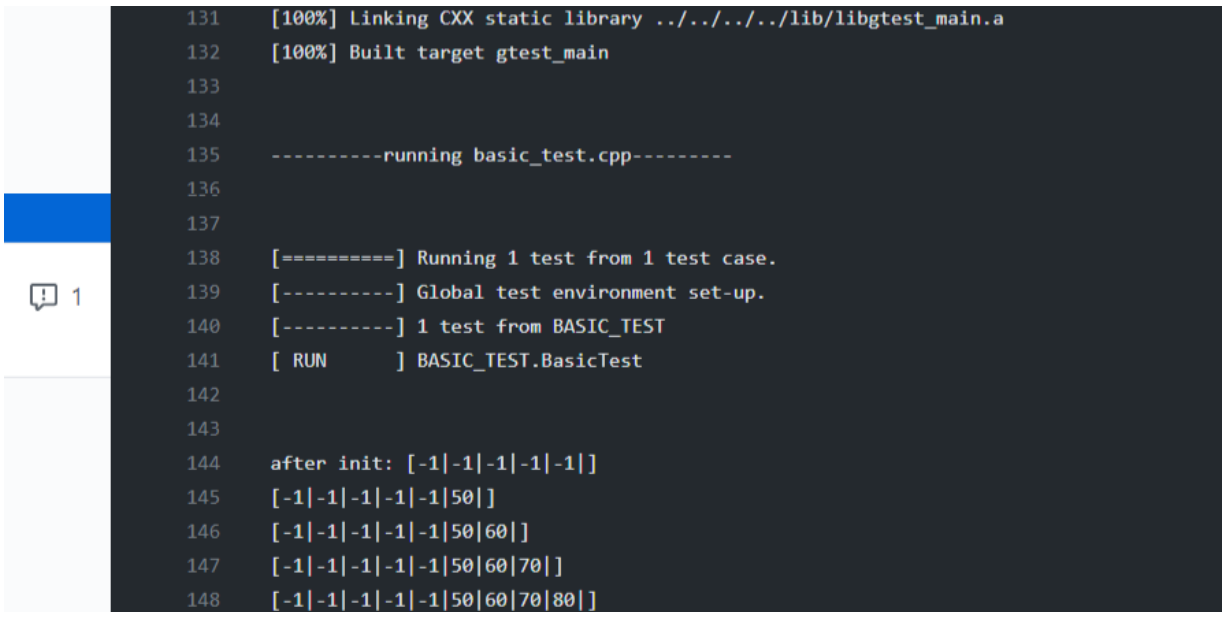
It will take you to this page:



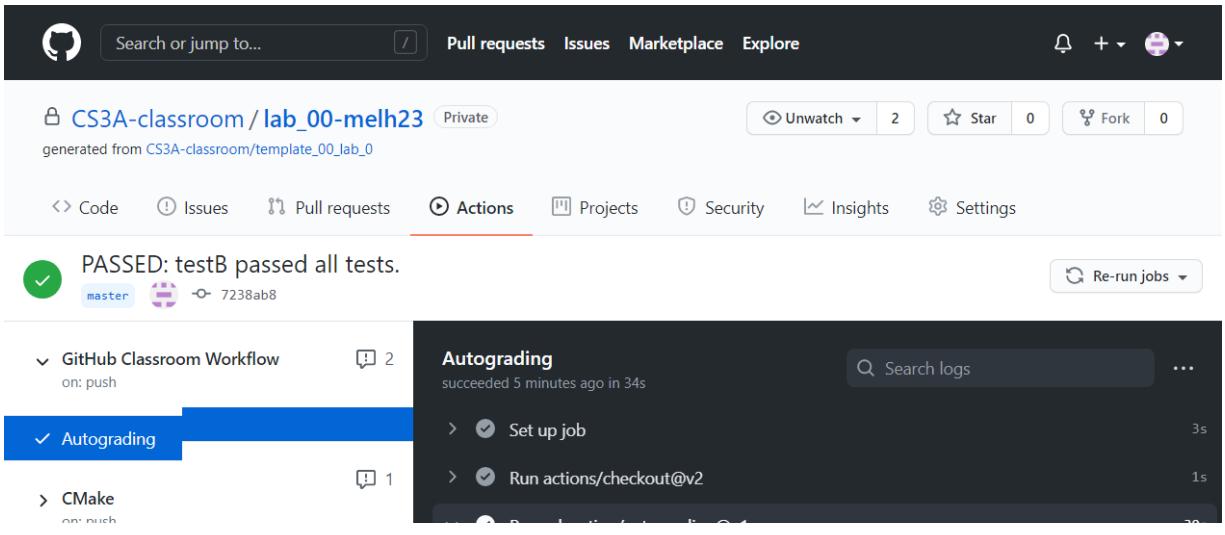

PASSED: testB passed all tests.
Autograding
succeeded 7 minutes ago in 34s
Run education/autograding@v1
[RUN] TEST_ARRAY.TestAt
[OK] TEST_ARRAY.TestAt (0 ms)
[----------] 3 tests from TEST_ARRAY (0 ms total)
[----------] Global test environment tear-down
[==========] 4 tests from 2 test cases ran. (1 ms total)
[PASSED] 4 tests.
testB
::1e41c4b3-f766-4da4-999e-293106bf9ee5::
All tests passed
Points 50/50
Post Run actions/checkout@v2
Complete job

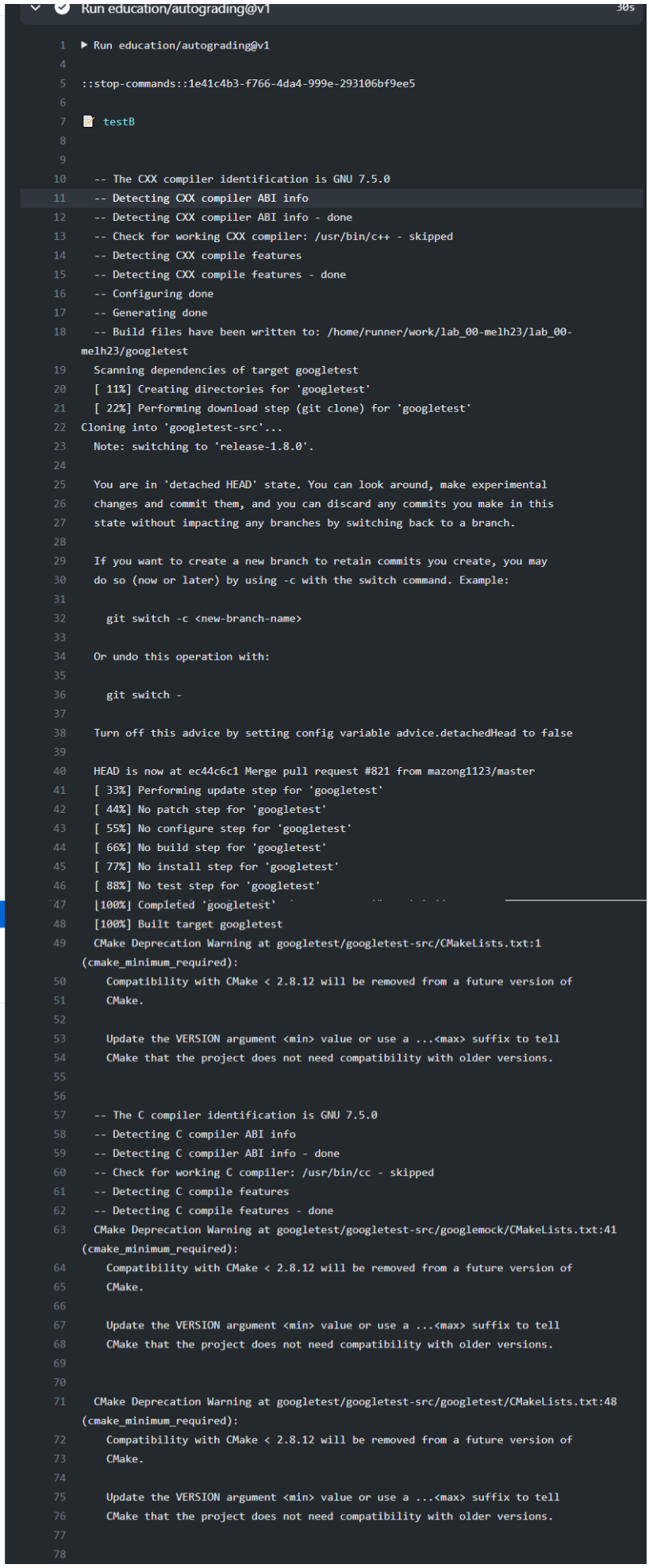

Search or jump to...
Pulls
Issues
Marketplace
Explore
CS3A-classroom / lab_00-melh23
Private
generated from CS3A-classroom/template_00_lab_0
Unwatch 2
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
master
Go to file
Add file
Code
melh23 added .bat files ...
6 minutes ago
4
.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
CS3A-classroom / lab_00-melh23
Private
Unwatch 2
Star 0
Fork 0
generated from CS3A-classroom/template_00_lab_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
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%


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|-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|]
1

```
[-1|-1|-1|-1|-1|50|60|70|80|90|]
found 70 at: 70
changing 70 to 700:
[-1|-1|-1|-1|-1|50|60|700|80|90|]


[       OK ] BASIC_TEST.BasicTest (0 ms)
[----------] 1 test from BASIC_TEST (0 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (0 ms total)
[  PASSED  ] 1 test.


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


[==========] Running 4 tests from 2 test cases.
[----------] 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 (0 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 (0 ms total)

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

✅ testB

```




Run education/autograding@v1
30s
▶ Run education/autograding@v1
::stop-commands::1e41c4b3-f766-4da4-999e-293106bf9ee5
testB
-- The CXX compiler identification is GNU 7.5.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/runner/work/lab_00-melh23/lab_00-melh23/googletest
Scanning dependencies of target googletest
[11%] Creating directories for 'googletest'
[22%] Performing download step (git clone) for 'googletest'
Cloning into 'googletest-src'...
Note: switching to 'release-1.8.0'.
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:
git switch -c <new-branch-name>
Or undo this operation with:
git switch -
Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at ec44c6c1 Merge pull request #821 from mazong1123/master
[33%] Performing update step for 'googletest'
[44%] No patch step for 'googletest'
[55%] No configure step for 'googletest'
[66%] No build step for 'googletest'
[77%] No install step for 'googletest'
[88%] No test step for 'googletest'
[100%] Completed 'googletest'
[100%] Built target googletest
CMake Deprecation Warning at googletest/googletest-src/CMakeLists.txt:1 (cmake_minimum_required):
Compatibility with CMake < 2.8.12 will be removed from a future version of
CMake.
Update the VERSION argument <min> value or use a ...<max> suffix to tell
CMake that the project does not need compatibility with older versions.
-- The C compiler identification is GNU 7.5.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
CMake Deprecation Warning at googletest/googletest-src/googlemock/CMakeLists.txt:41 (cmake_minimum_required):
Compatibility with CMake < 2.8.12 will be removed from a future version of
CMake.
Update the VERSION argument <min> value or use a ...<max> suffix to tell
CMake that the project does not need compatibility with older versions.
CMake Deprecation Warning at googletest/googletest-src/googletest/CMakeLists.txt:48 (cmake_minimum_required):
Compatibility with CMake < 2.8.12 will be removed from a future version of
CMake.
Update the VERSION argument <min> value or use a ...<max> suffix to tell
CMake that the project does not need compatibility with older versions.

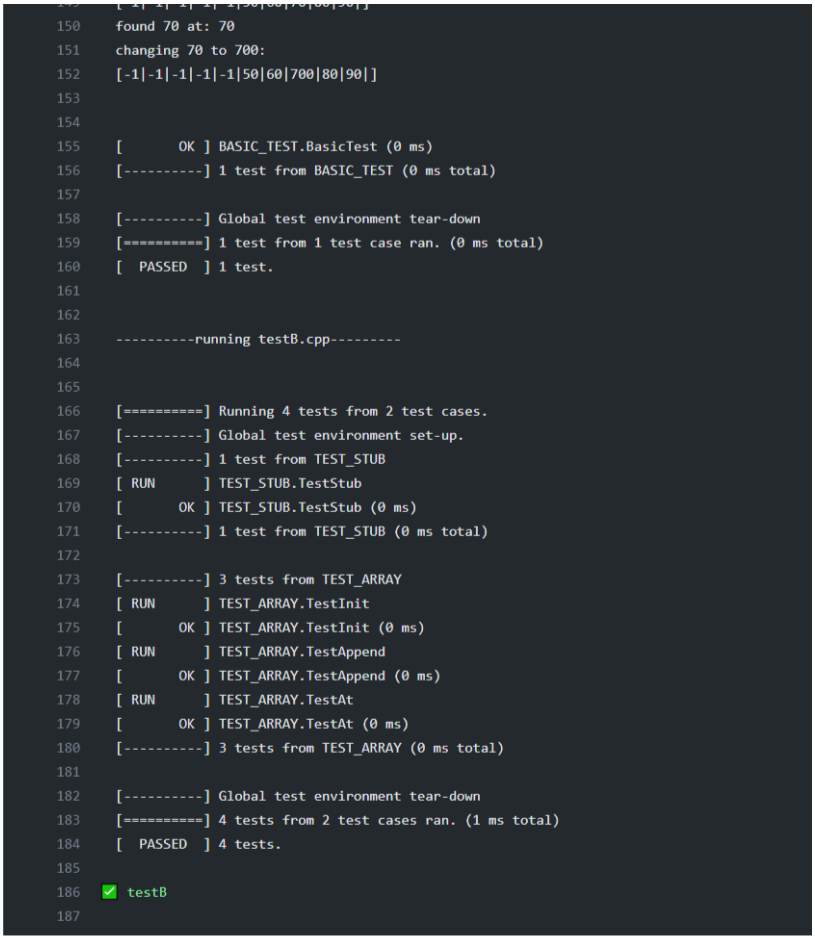
```
-- Found PythonInterp: /usr/bin/python (found version "2.7.17")
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Failed
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /home/runner/work/lab_00-melh23/lab_00-melh23

Scanning dependencies of target main
[  4%] Building CXX object CMakeFiles/main.dir/main.cpp.o
[  8%] Building CXX object CMakeFiles/main.dir/includes/stub/stub.cpp.o
```

1

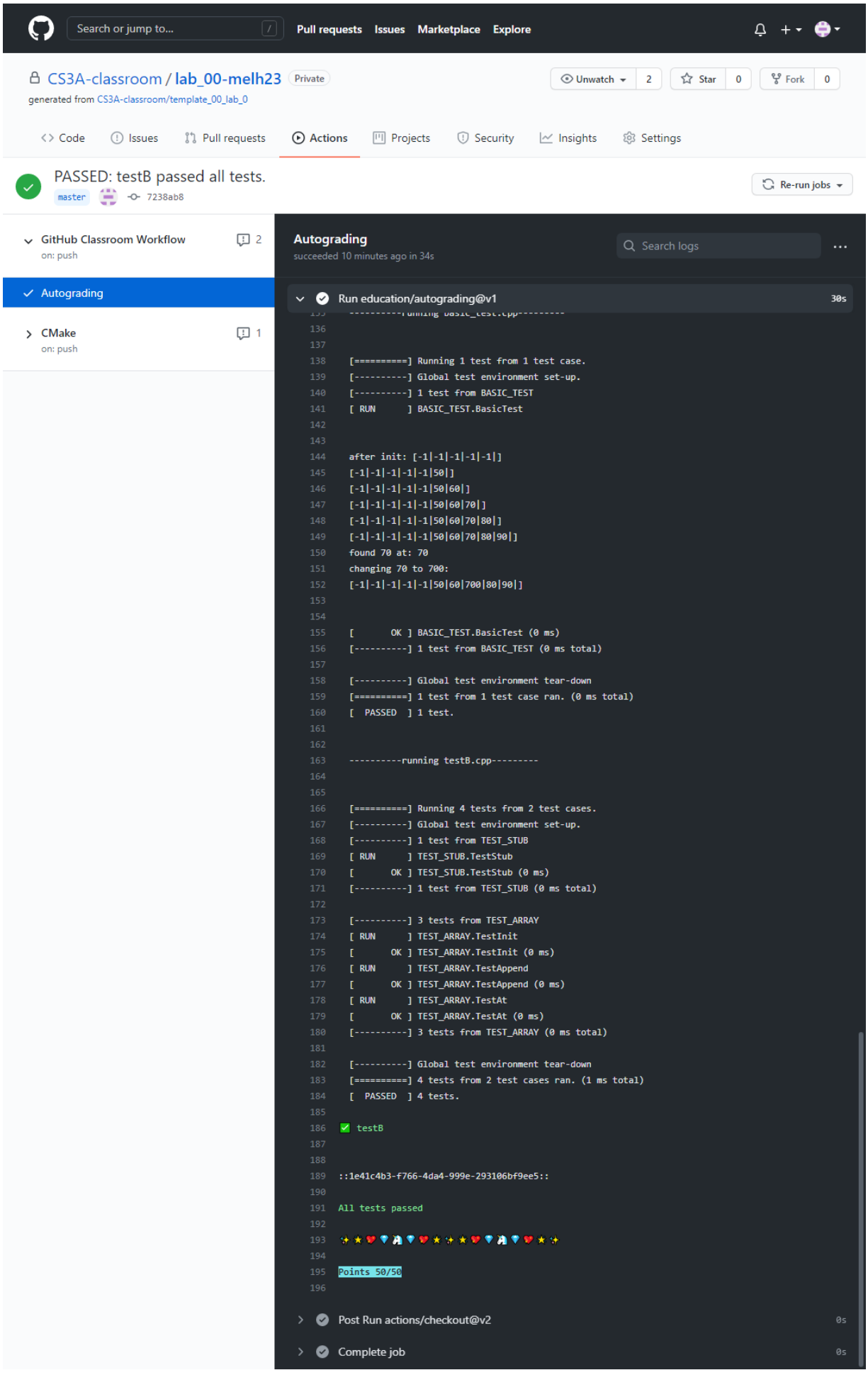
```
[ 12%] Linking CXX executable bin/main
[ 12%] Built target main
Scanning dependencies of target gtest
[ 16%] Building CXX object googletest/googletest-build/googlemock/gtest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 20%] Linking CXX static library ../../../../lib/libgtest.a
[ 20%] Built target gtest
Scanning dependencies of target basic_test
[ 25%] Building CXX object CMakeFiles/basic_test.dir/_tests/_test_files/basic_test.cpp.o
[ 29%] Building CXX object CMakeFiles/basic_test.dir/includes/array_functions/array_functions.cpp.o
[ 33%] Linking CXX executable bin/basic_test
[ 33%] Built target basic_test
Scanning dependencies of target testA
[ 37%] Building CXX object CMakeFiles/testA.dir/_tests/_test_files/testA.cpp.o
[ 41%] Building CXX object CMakeFiles/testA.dir/includes/array_functions/array_functions.cpp.o
[ 45%] Linking CXX executable bin/testA
[ 45%] Built target testA
Scanning dependencies of target testB
[ 50%] Building CXX object CMakeFiles/testB.dir/_tests/_test_files/testB.cpp.o
[ 54%] Building CXX object CMakeFiles/testB.dir/includes/array_functions/array_functions.cpp.o
[ 58%] Building CXX object CMakeFiles/testB.dir/includes/stub/stub.cpp.o
[ 62%] Linking CXX executable bin/testB
[ 62%] Built target testB
Scanning dependencies of target gmock_main
[ 66%] Building CXX object googletest/googletest-build/googlemock/CMakeFiles/gmock_main.dir/__/googletest/src/gtest-all.cc.o
[ 70%] Building CXX object googletest/googletest-build/googlemock/CMakeFiles/gmock_main.dir/src/gmock-all.cc.o
[ 75%] Building CXX object googletest/googletest-build/googlemock/CMakeFiles/gmock_main.dir/src/gmock_main.cc.o
[ 79%] Linking CXX static library ../../../lib/libgmock_main.a
[ 79%] Built target gmock_main
Scanning dependencies of target gmock
[ 83%] Building CXX object googletest/googletest-build/googlemock/CMakeFiles/gmock.dir/__/googletest/src/gtest-all.cc.o
[ 87%] Building CXX object googletest/googletest-build/googlemock/CMakeFiles/gmock.dir/src/gmock-all.cc.o
[ 91%] Linking CXX static library ../../../lib/libgmock.a
[ 91%] Built target gmock
Scanning dependencies of target gtest_main
[ 95%] Building CXX object googletest/googletest-build/googlemock/gtest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.o
[100%] Linking CXX static library ../../../../lib/libgtest_main.a
[100%] Built target gtest_main


----------running basic_test.cpp---------
```

1

```
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from BASIC_TEST
[ RUN      ] BASIC_TEST.BasicTest


after init: [-1|-1|-1|-1|-1|]
[-1|-1|-1|-1|-1|50|]
[-1|-1|-1|-1|-1|50|60|]
[-1|-1|-1|-1|-1|50|60|70|]
[-1|-1|-1|-1|-1|50|60|70|80|]
[-1|-1|-1|-1|-1|50|60|70|80|90|]
```



```
found 70 at: 70
changing 70 to 700:
[-1|-1|-1|-1|-1|50|60|700|80|90|]


[       OK ] BASIC_TEST.BasicTest (0 ms)
[----------] 1 test from BASIC_TEST (0 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (0 ms total)
[  PASSED  ] 1 test.


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


[==========] Running 4 tests from 2 test cases.
[----------] 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 (0 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 (0 ms total)

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

✅ testB
```


Search or jump to...
Pull requests
Issues
Marketplace
Explore
CS3A-classroom / lab_00-melh23
Private
generated from CS3A-classroom/template_00_lab_0
Unwatch 2
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
PASSED: testB passed all tests.
master
7238ab8
Re-run jobs
GitHub Classroom Workflow
on: push
2
Autograding
CMake
on: push
1
Autograding
succeeded 10 minutes ago in 34s
Search logs
Run education/autograding@v1
30s
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from BASIC_TEST
[RUN] BASIC_TEST.BasicTest
after init: [-1|-1|-1|-1|-1|]
[-1|-1|-1|-1|-1|50|]
[-1|-1|-1|-1|-1|50|60|]
[-1|-1|-1|-1|-1|50|60|70|]
[-1|-1|-1|-1|-1|50|60|70|80|]
[-1|-1|-1|-1|-1|50|60|70|80|90|]
found 70 at: 70
changing 70 to 700:
[-1|-1|-1|-1|-1|50|60|700|80|90|]
[OK] BASIC_TEST.BasicTest (0 ms)
[----------] 1 test from BASIC_TEST (0 ms total)
[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (0 ms total)
[PASSED] 1 test.
----------running testB.cpp---------
[==========] Running 4 tests from 2 test cases.
[----------] 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 (0 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 (0 ms total)
[----------] Global test environment tear-down
[==========] 4 tests from 2 test cases ran. (1 ms total)
[PASSED] 4 tests.
testB
::1e41c4b3-f766-4da4-999e-293106bf9ee5::
All tests passed
Points 50/50
Post Run actions/checkout@v2
0s
Complete job
0s