

CS3A-classroom / lab0_writeup

<> Code Issues Pull requests Actions Projects Wiki Security

master ▾

...

lab0_writeup / mac.md

 barkeshli added CS3, CS8 links

🕒 History

👤 1 contributor

Raw Blame



789 lines (407 sloc) | 27.5 KB

Mac Instructions

- [Installing cmake](#)
- [Accepting the assignment](#)
- [Project organization](#)
- [Quick edit, status , add , commit , & push](#)
- [Getting started with the project](#)
- [Writing tests](#)
- [Completing the project](#)

■ 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.

```
CS-xx-projects 🍏$> cmake --version
cmake version 3.16.0-rc4

CMake suite maintained and supported by Kitware (kitware.com/cmake).
CS-xx-projects 🍏$>
```

install cmake

We use homebrew to install cmake :

```
brew install cmake
```

This will go on and on...

```
CS-xx-projects 🍏$> brew install cmake
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 4 taps (homebrew/core, homebrew/cask, homebrew/services and mongodb/brew).
==> New Formulae
aerc                                libxcomposite
aida-header                           libxcursor
aliddns                             libxdamage
ansible@2.9                            libxdmcp
arturo                               libxext
asroute                               libxfixes
atkmm@2.28                            libxfont
attr                                 libxft
aws-console                           libxi
aws-rotate-key                        libxinerama
bandit                                libxkbfile
```

... and on...

```
==> Updated Formulae
Updated 4905 formulae.
==> Renamed Formulae
glibmm@2.64 -> glibmm@2.66          pangomm@2.42 -> pangomm@2.46
gtk+4 -> gtk4                         prest -> prestd
now-cli -> vercel-cli
==> Deleted Formulae
boost@1.55                                godep           scw@1
boost@1.59                                llvm@6          stlviewer
confluent-platform                         meson-internal  unp64
curl-openssl                            mysql-connector-c++@1.1  unrar
dtrx                                     ori              woboq_codebrowser
fmsx                                     pgplot          xspin
gobby                                    rmtrash         xu4
==> New Casks
7777
abysssoft-teleport
accordance
aio-creator-neo
aldente
aleo-studio
amazon-workdocs-drive
anka-build-cloud-controller-and-registry
anka-build-cloud-registry
anka-virtualization
atemosc
aural
```

... and on...

```
Warning: Treating cmake as a formula. For the cask, use homebrew/cask/cmake
==> Downloading https://homebrew.bintray.com/bottles/cmake-3.19.4.mojave.bottle
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/c4eac1fa4580a117a33
0.7
#
## 2.3
### 3.8
#### 5.5
##### 6.9
###### 8.6
####### 10.2
######## 11.8
######### 13.2
########## 14.9
########## 16.6
########### 18.2
########## 19.9
########## 21.4
########## 23.0
########## 24.6
########## 26.1
########## 27.7
########## 29.4
########## 30.9
########## 32.6
########## 34.1
########## 35.7
```

... and on...

Finally, cmake is installed!

```
--> Summary
🍺 /usr/local/Cellar/cmake/3.19.4: 6,376 files, 63.9MB
--> `brew cleanup` has not been run in 30 days, running now...
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/gdbm--1.18.1_1.mojave.bottle.tar.gz... (203.3KB)
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/openssl@1.1--1.1.1g.mojave.bottle.tar.gz... (5.3MB)
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/python@3.8--3.8.5.mojave.bottle.tar.gz... (16.6MB)
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/readline--8.0.4.mojave.bottle.tar.gz... (517.6KB)
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/sqlite--3.33.0.mojave.bottle.tar.gz... (2.0MB)
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/xz--5.2.5.mojave.bottle.tar.gz... (386.8KB)
Removing: /Users/sassanbarkeshli/Library/Caches/Homebrew/portable-ruby-2.6.3_2.yosemite.bottle.tar.gz... (9.1MB)
Removing: /Users/sassanbarkeshli/Library/Logs/Homebrew/gdbm... (64B)
Removing: /Users/sassanbarkeshli/Library/Logs/Homebrew/readline... (64B)
Removing: /Users/sassanbarkeshli/Library/Logs/Homebrew/sqlite... (64B)
Removing: /Users/sassanbarkeshli/Library/Logs/Homebrew/xz... (64B)
Removing: /Users/sassanbarkeshli/Library/Logs/Homebrew/openssl@1.1... (64B)
Removing: /Users/sassanbarkeshli/Library/Logs/Homebrew/python@3.8... (3 files, 172.3KB)
Pruned 1 symbolic links and 2 directories from /usr/local
```

check the version of the cmake again:

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

```
CS-xx-projects 🍏 $> cmake --version
cmake version 3.16.0-rc4

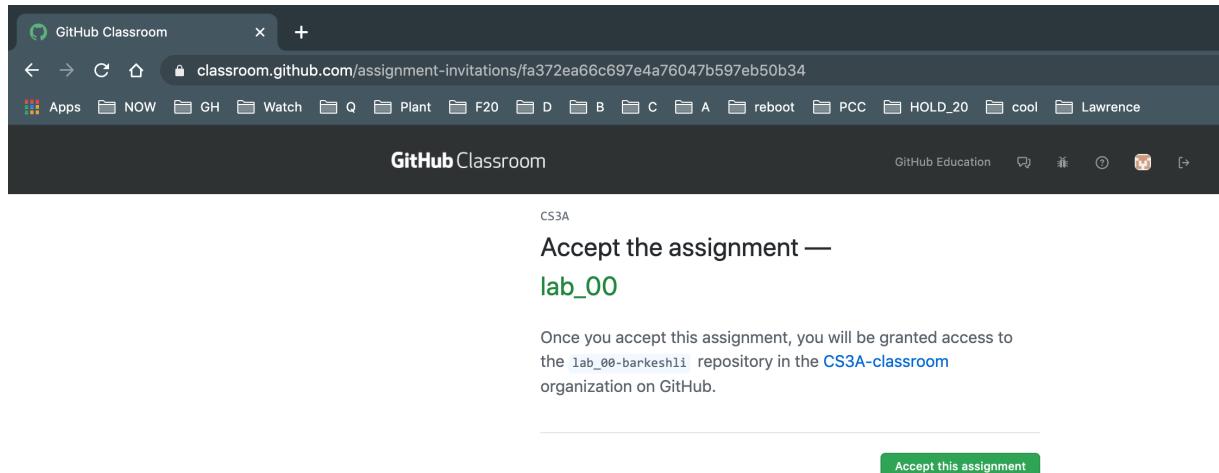
CMake suite maintained and supported by Kitware (kitware.com/cmake).
CS-xx-projects 🍏 $>
```

■ 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 assignment?

the lab_00-barkeshli repository in the CS3A-classroom organization on GitHub

Accept the assignment

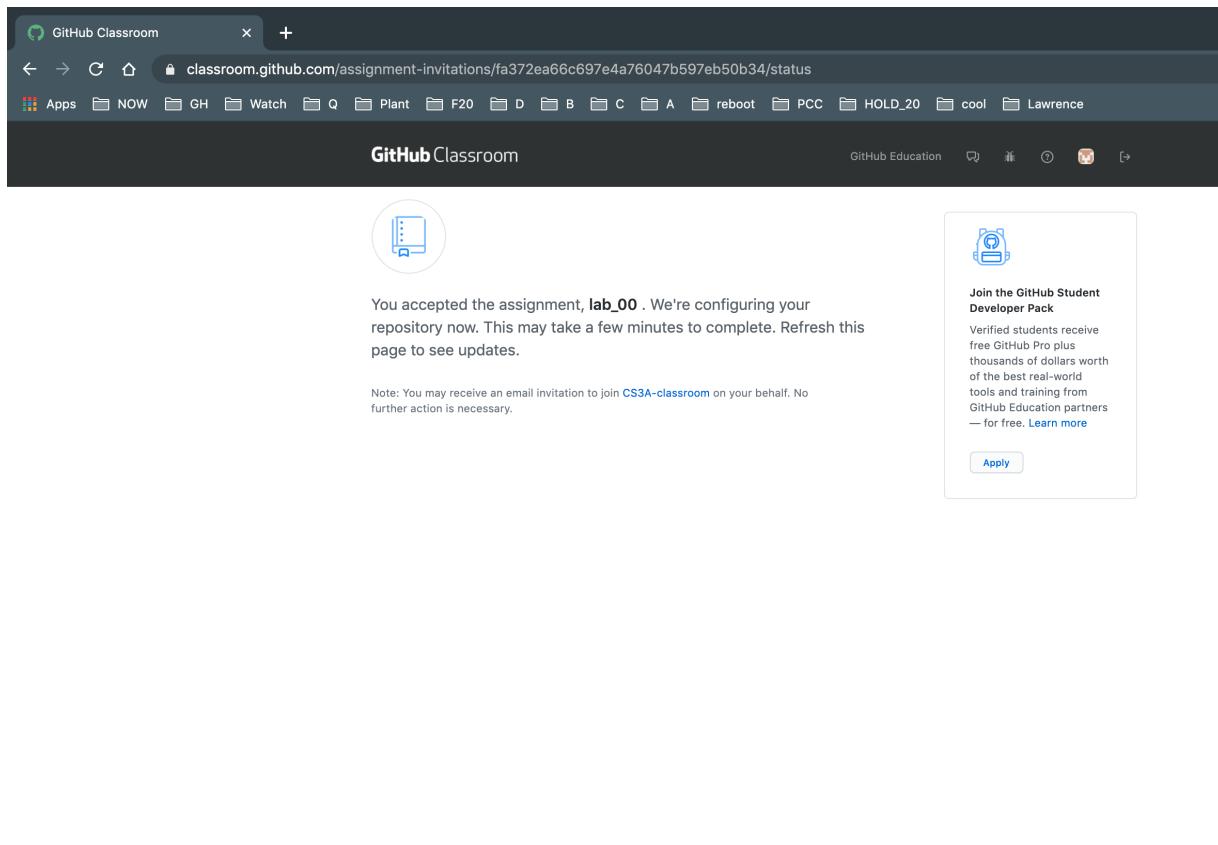
You will *accept* the assignment by clicking the green button.

After you accept the assignment, you will be granted access to the repository 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." link and an "Apply" button.

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!" indicating acceptance of the assignment "lab_00". It also shows that a repository has been created at https://github.com/CS3A-classroom/lab_00-barkeshli. A note states: "We've configured the repository associated with this assignment ([update](#))."

look closer:

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

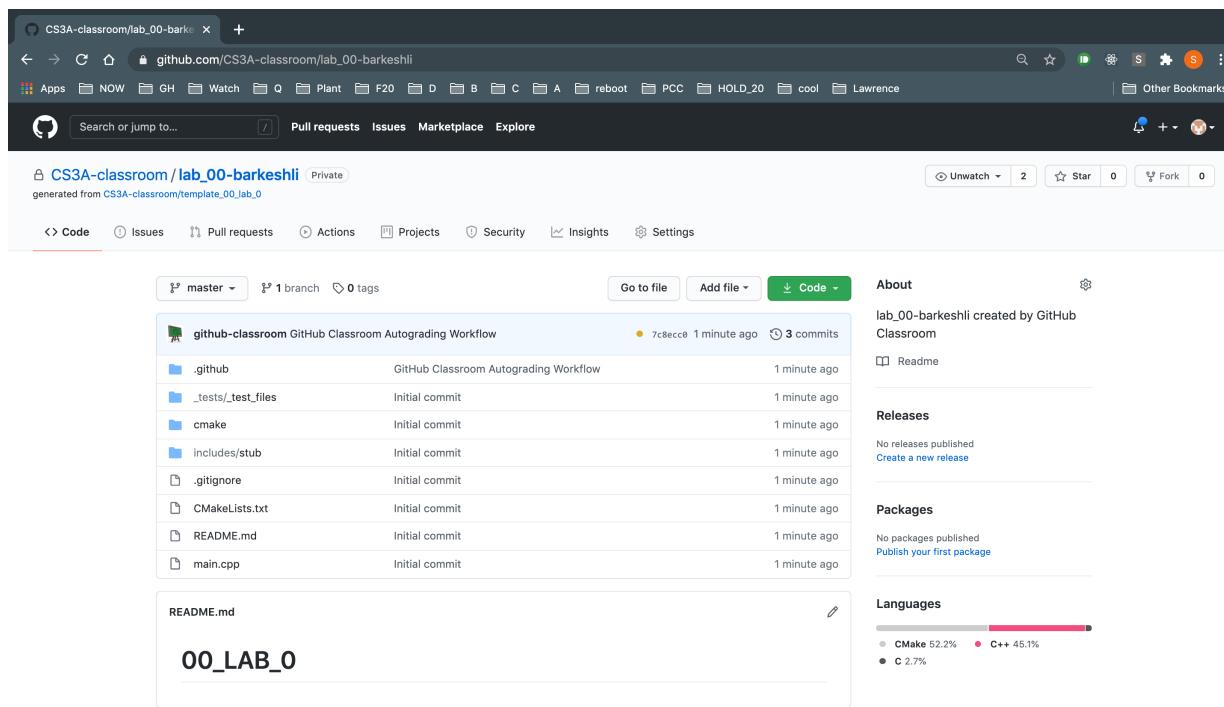
YOUR ASSIGNMENT REPOSITORY HAS BEEN CREATED.

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:

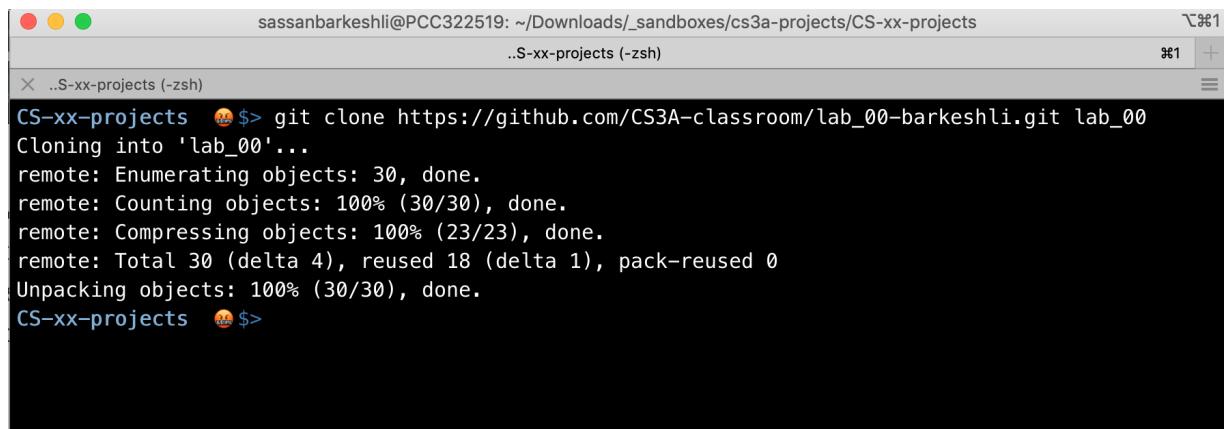
Before you can work on your project, you will need a local copy of the assignment. This is called **cloning** the repository.

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.

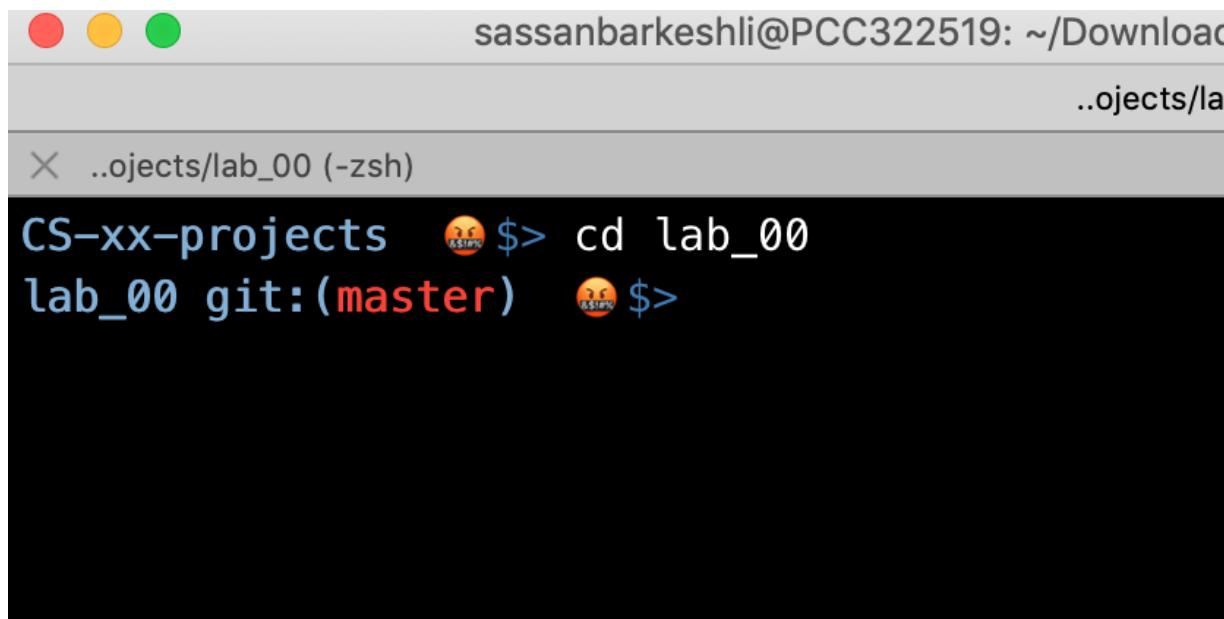
and then: `git clone [clone link] [destination_folder]`

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`



sassanbarkeshli@PCC322519: ~/Downloads/_sandboxes/cs3a-projects/CS-xx-projects
..S-xx-projects (-zsh)
CS-xx-projects 😊 \$> 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 😊 \$>

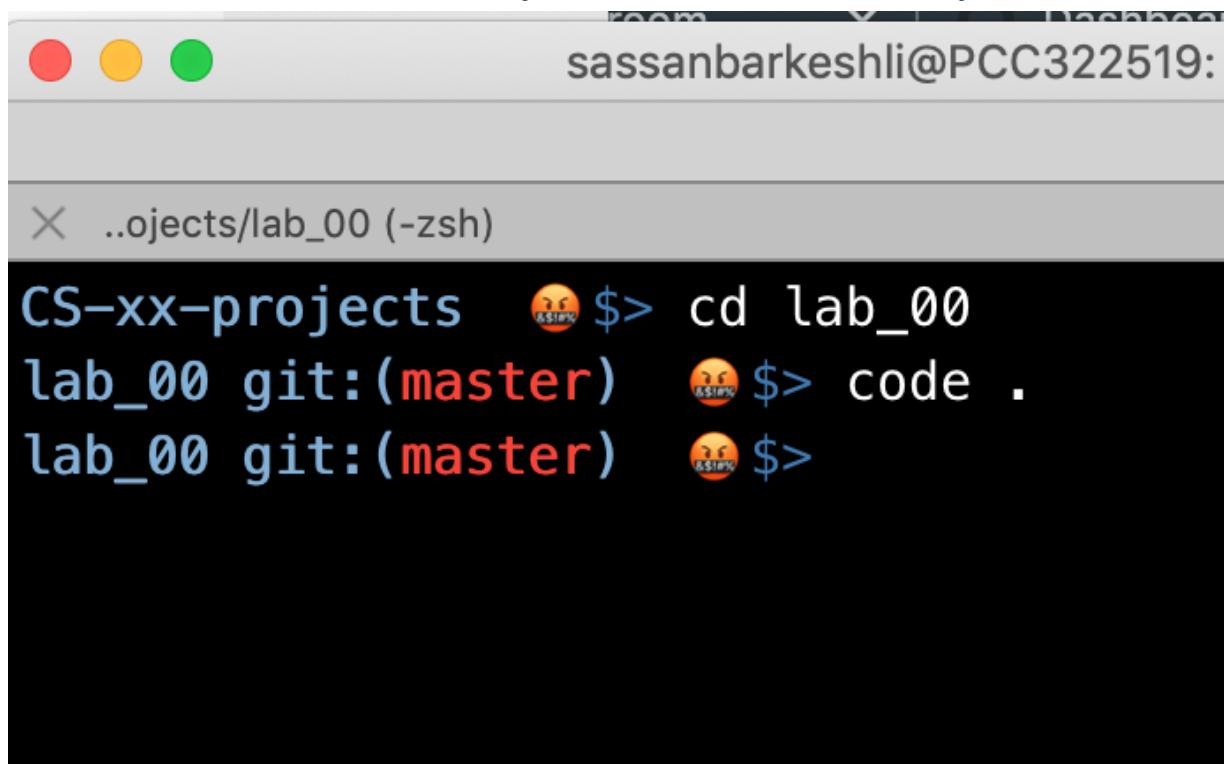
cd into this project folder:



sassanbarkeshli@PCC322519: ~/Downloads
..objects/la
X ..objects/lab_00 (-zsh)
CS-xx-projects 😊 \$> cd lab_00
lab_00 git:(master) 😊 \$>

code . :

this will open VSCode and loads the current folder into it.

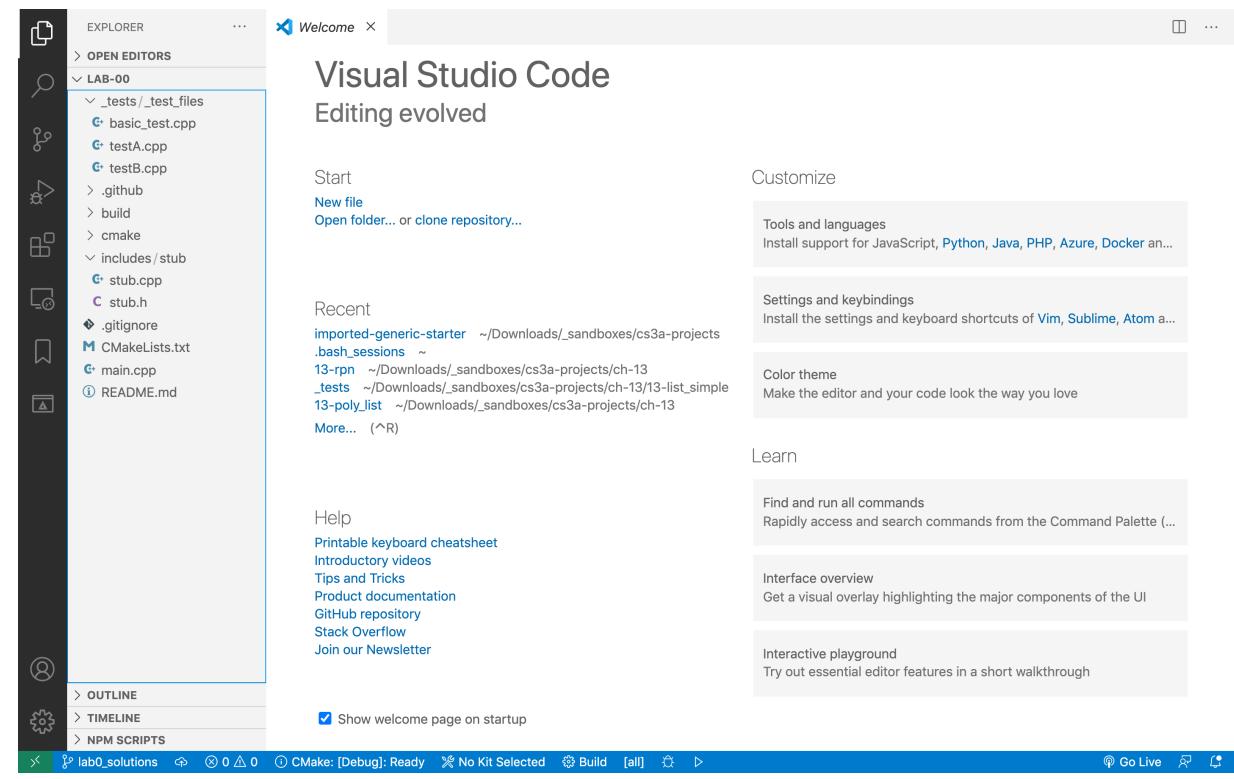


A screenshot of a macOS terminal window. The title bar shows the user's name, sassanbarkeshli@PCC322519: . Below the title bar, there is a tab labeled "X ..objects/lab_00 (-zsh)". The main area of the terminal shows the following command sequence:

```
CS-xx-projects $> cd lab_00
lab_00 git:(master) $> code .
lab_00 git:(master) $>
```

■ Project Organization: ■

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



File system:

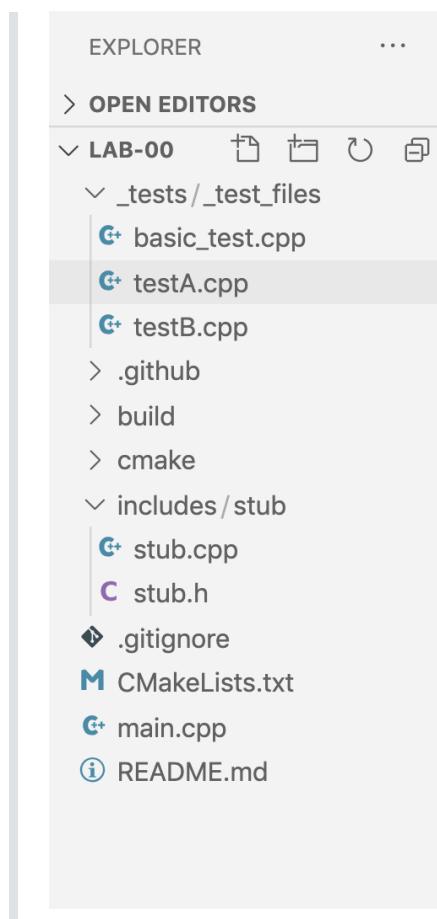
All the projects in this class will follow the same file organization.

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
    OUTLINE
    TIMELINE
    NPM SCRIPTS
    lab0_solutions
    0 ▲ 0 ① 4
    CMake: [Debug] Ready
    No Kit Selected
    Build [all]
    Ln 1, Col 1 Spaces: 2 UTF-8 LF C++ Go Live Mac
    https://github.com/CS3A-classroom/lab0_writeup/blob/master/mac.md

```

```

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 //-----COPY BASIC_TEST INTO THIS FILE.
7 // AND THEN,
8 // DO NOT EDIT THIS FILE ANY FURTHER
9 //-----
10 //-----TEST(BASIC_TEST, BasicTest) {
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();

```

```

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

```

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...)

```

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

```

The screenshot shows the VS Code interface with the CMakeLists.txt file open in the center editor. The left sidebar shows the project structure with files like basic_test.cpp, testA.cpp, testB.cpp, stub.cpp, and stub.h. The bottom status bar shows build information and file details.

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.

```

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

```

```

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

```

The screenshot shows the VS Code interface with two files open: stub.h and stub.cpp. The stub.h file contains a header guard and a definition for the stub() function. The stub.cpp file contains the implementation of stub() which returns true. The left sidebar shows the project structure with files like basic_test.cpp, testA.cpp, testB.cpp, stub.cpp, and stub.h.

■ 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', 'CMakeLists.txt', 'main.cpp', and 'README.md'. The 'README.md' file is currently open in the main editor area, displaying the text '# 00_LAB_0'. Below the editor is the Terminal panel, which shows the output of a 'git status' command:

```
lab_00 git:(master) ✘ $ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
lab_00 git:(master) ✘ $
```

The status bar at the bottom indicates the current workspace is 'master'.

Edit the README.md

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

The screenshot shows a GitHub code editor interface. At the top, there's a header bar with icons for file operations like copy, paste, and close. Below the header, the file path is shown as "README.md > abc # 00_LAB_0". The main content area displays two lines of text:

```
1 # 00_LAB_0
2 Sassan Barkeshli
```

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.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in the 'LAB_00' folder: '_tests', '.github', 'cmake', 'includes', '.gitignore', 'CMakeLists.txt', 'main.cpp', and 'README.md'. The 'README.md' file is open in the main editor area, showing the following content:

```
① README.md x
① README.md > # 00_LAB_0
1 # 00_LAB_0
2
3 Sassan Barkeshli
4
```

The terminal at the bottom shows the following git status output:

```
TERMINAL    PROBLEMS    OUTPUT    DEBUG CONSOLE
1: zsh      +  ×  ^  ×
lab_00 git:(master) ✘  🐧 $> 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 checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
lab_00 git:(master) ✘  🐧 $>
```

The status bar at the bottom indicates: master*, 0 △ 0, CMake: [Debug]: Ready, No Kit Selected, Build [all], Go Live, ESLint, Prettier.

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 VS Code interface. In the Explorer sidebar, there's a folder named 'LAB_00' containing files like '_tests', '.github', 'cmake', 'includes', '.gitignore', 'CMakeLists.txt', 'main.cpp', and 'README.md'. The 'README.md' file is open in the main editor area, showing a commit message:

```

1 # 00_LAB_0
2
3 Sassan Barkeshli
4

```

Below the editor is a terminal window titled 'zsh' showing the output of a 'git status' command:

```

lab_00 git:(master) ✘ 🎉$ 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 checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
lab_00 git:(master) ✘ 🎉$ git add README.md
lab_00 git:(master) ✘ 🎉$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md

lab_00 git:(master) ✘ 🎉$ 

```

The bottom status bar shows various icons and text, including 'CMake: [Debug]: Ready', 'No Kit Selected', 'Build [all]', and 'Prettier: ✓'.

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.

```

# 00_LAB_0
Sassan Barkeshli

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 checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
lab_00 git:(master) ✘ $ git add README.md
lab_00 git:(master) ✘ $ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md

lab_00 git:(master) ✘ $ git commit -m "added name to README.md"
[master 7fafd0c] added name to README.md
 1 file changed, 2 insertions(+)
lab_00 git:(master) ✘ $ 

```

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!

Committing only records your changes on your local machine.

About

lab_00-barkeshli created by GitHub Classroom

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%

What is my local branch name?

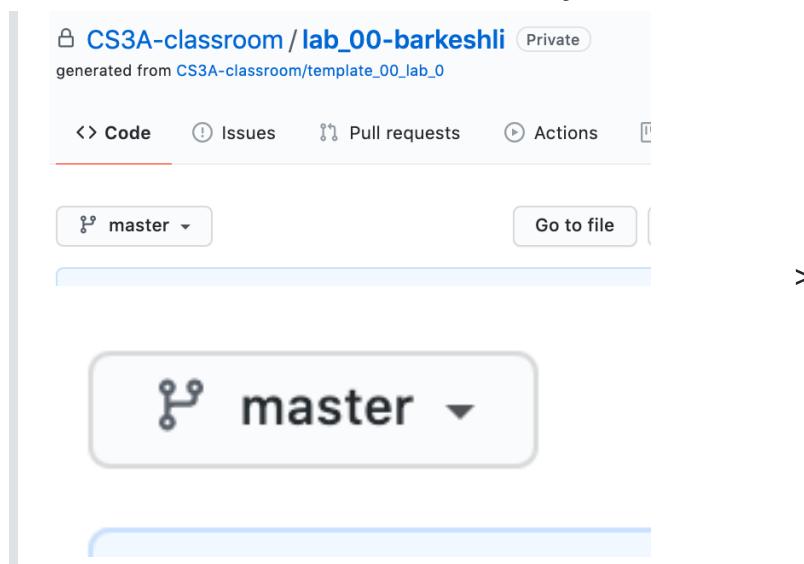
If we want Github to know about our changes, we must push them to Github. But before you do, let's find out what our current *branch* is.

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

Your bash prompt on your terminal:

`lab_00 git:(master)`

the branch name on your Github repo page:



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 command will be `git push origin main` :

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

```
nothing to commit, working tree clean
lab_00 git:(master) 🎉 $> git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/CS3A-classroom/lab_00-barkeshli.git
  7c8ecc0..7fafd0c  master -> master
lab_00 git:(master) 🎉 $>
```

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?

■ 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.

```

_explorer
... _tests > _test_files > basic_test.cpp > ...
basic_test.cpp 2, M
...
3 #include <iomanip>
4 #include "../includes/stub/stub.h"
5 #include "../includes/array_functions/array_functions.h"
6 //-----
7 //      COPY BASIC_TEST INTO THIS FILE.
8 //          AND THEN,
9 //      DO NOT EDIT THIS FILE ANY FURTHER
10 //-----
11
12 bool test_stub(){
13     bool test = stub();
14     return test;
15 }
16 bool test_append(bool debug=false){
17     const int MAX = 20;
18     int a[MAX];
19     int size = 5;
20     const char tabs[] = "\t-----\t";
21     if (debug)
22     {
23         cout << tabs<<"size: 5, init array to -1" << endl;
24     }
25     _array_init(a, size, -1);
26     if (debug){
27         cout << tabs;
28         if(debug) _print_array(a, size);
29         cout << endl;
30     }
31     for (int i = size; i < 10; i++){
32         _append(a, size, i * 10);
33         if(debug){
34             cout << tabs;
35             _print_array(a, size);
36             cout << endl;
37         }
38     }
39     if (size!=10){
40         cout << "FAILED: Expected size to be 10, but found " << size << endl;
41         return false;
}
...
Ln 1, Col 1 Spaces: 2 UTF-8 LF C++ ⌂ Go Live Mac ⌂ ⌂

```

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

name this folder `array_functions`.

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

```

basic_test.cpp
...
#include <iomanip>
#include "../includes/stub/stub.h"
#include "../includes/array_functions/array_functions.h"
// COPY BASIC_TEST INTO THIS FILE.
// AND THEN,
// DO NOT EDIT THIS FILE ANY FURTHER
bool test_stub(){
    bool test = stub();
    return test;
}
bool test_append(bool debug=false){
    const int MAX = 20;
    int a[MAX];
    int size = 5;
    const char tabs[] = "\t-----\t";
    if (debug)
    {
        cout << tabs<<"size: 5, init array to -1" << endl;
    }
    _array_init(a, size, -1);
    if (debug)
    {
        cout << tabs;
        _print_array(a, size);
        cout << endl;
    }
    for (int i = size; i < 10; i++){
        _append(a, size, i * 10);
        if(debug)
        {
            cout << tabs;
            _print_array(a, size);
            cout << endl;
        }
    }
    if (size!=10){
        cout << "FAILED: Expected size to be 10, but found " << size << endl;
        return false;
    }
}

testA.cpp
...
#include "gtest/gtest.h"
#include <iostream>
#include <iomanip>
#include "../includes/stub/stub.h"
using namespace std;
// COPY BASIC_TEST INTO THIS FILE.
// AND THEN,
// DO NOT EDIT THIS FILE ANY FURTHER
bool basic_test(bool debug = true){
    if (debug){
        cout << "\nbasic test...\n" << endl;
    }
    return true;
}
TEST(BASIC_TEST, BasicTest) {
    //EXPECT_EQ(1, <your individual test function>);
    EXPECT_EQ(1, basic_test(false));
}
int main(int argc, char **argv) {
    ::testing::InitGoogleTest(&argc, argv);
    std::cout<<"\n\n-----running basic_tests-----";
    return RUN_ALL_TESTS();
}

```

Add two files to this folder.

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

```

basic_test.cpp
...
#include <iomanip>
#include "../includes/stub/stub.h"
#include "../includes/array_functions/array_functions.h"
// COPY BASIC_TEST INTO THIS FILE.
// AND THEN,
// DO NOT EDIT THIS FILE ANY FURTHER
bool test_stub(){
    bool test = stub();
    return test;
}
bool test_append(bool debug=false){
    const int MAX = 20;
    int a[MAX];
    int size = 5;
    const char tabs[] = "\t-----\t";
    if (debug)
    {
        cout << tabs<<"size: 5, init array to -1" << endl;
    }
    _array_init(a, size, -1);
    if (debug)
    {
        cout << tabs;
        _print_array(a, size);
        cout << endl;
    }
    for (int i = size; i < 10; i++){
        _append(a, size, i * 10);
        if(debug)
        {
            cout << tabs;
            _print_array(a, size);
            cout << endl;
        }
    }
    if (size!=10){
        cout << "FAILED: Expected size to be 10, but found " << size << endl;
        return false;
    }
}

testA.cpp
...
#include "gtest/gtest.h"
#include <iostream>
#include <iomanip>
#include "../includes/stub/stub.h"
using namespace std;
// COPY BASIC_TEST INTO THIS FILE.
// AND THEN,
// DO NOT EDIT THIS FILE ANY FURTHER
bool basic_test(bool debug = true){
    if (debug){
        cout << "\nbasic test...\n" << endl;
    }
    return true;
}
TEST(BASIC_TEST, BasicTest) {
    //EXPECT_EQ(1, <your individual test function>);
    EXPECT_EQ(1, basic_test(false));
}
int main(int argc, char **argv) {
    ::testing::InitGoogleTest(&argc, argv);
    std::cout<<"\n\n-----running basic_tests-----";
    return RUN_ALL_TESTS();
}

TERMINAL
lab-00 git:(lab0_solutions) ✘ $ git checkout solutions _tests/_test_files/basic_test.cpp
error: pathspec 'solutions' did not match any file(s) known to git
lab-00 git:(lab0_solutions) ✘ $ git checkout solution _tests/_test_files/basic_test.cpp
lab-00 git:(lab0_solutions) ✘ $ git checkout solution _tests/_test_files/basic_test.cpp
lab-00 git:(lab0_solutions) ✘ $ git checkout solution includes/array_functions/array_functions.h
lab-00 git:(lab0_solutions) ✘ $ git checkout solution includes/array_functions/array_functions.cpp
lab-00 git:(lab0_solutions) ✘ $ 

```

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`

```
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
19
20 #endif // ARRAY_FUNCTIONS_H
21
```

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.

```

C 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 int _find(const int a[], int size, int find_me);
11 int& _at(int a[], int size, int pos);
12
13 ostream& _print_array(const int a[], int size, ostream& outs = cout);
14
15
16
17
18
19
20 #endif // ARRAY_FUNCTIONS_H
21

C array_functions.h
includes > array_functions > C array_functions.cpp > _print_array(const int a[], int, ostream& outs)
2  include "cassert"
3
4
5  void _array_init(int a[], int size, int x=0){
6
7  }
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
25 ostream& _print_array(const int a[], int size, ostream& outs = cout){
26
27     return outs;
28 }
29

```

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.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure with files like `array_functions.h`, `basic_test.cpp`, `testA.cpp`, `testB.cpp`, `CMakeLists.txt`, and `README.md`.
- TERMINAL:** Shows the command `git:(lab0_solutions) x $>`.
- STATUS BAR:** Shows the current file is `array_functions.h`, and other details like Ln 21, Col 5, Spaces: 4, UTF-8, LF, C++, Go Live, Mac.

Go to build/ and run cmake :

cd into the `build/` folder, and from there, run `cmake ..`

This will run `cmake` on your *parent* folder. (that's what `..` means. `cmake ..` means run `cmake` on my parent folder.) `cmake` creates a bunch of files and you do not want these files in the root folder of your project. Running `cmake` from `build/` will make sure all your auxiliary files are created inside the `build/` folder.

Hopefully, your `cmake` will run without any problems and it will tell you that "Build files are written to `.. . build/`"

Now, we are ready to compile our project using `make`

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure:
 - OPEN EDITORS: `array_functions.h`
 - LAB-00:
 - _tests/_test_files: `basic_test.cpp`
 - testA.cpp
 - testB.cpp
 - .github
 - build
 - cmake
 - includes:
 - array_functions:
 - `array_functions.c...` (selected)
 - `array_functions.h`
 - stub
 - `stub.cpp`
 - `stub.h`
 - .gitignore
 - CMakeLists.txt
 - main.cpp
 - README.md
- TERMINAL:** Displays the build process for 'lab-00':


```
Lab-00 git:(lab0_solutions) ✘ $ cd build
build git:(lab0_solutions) ✘ $ cmake ..
-- The CXX compiler identification is AppleClang 10.0.1.10010046
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/build/googletest
Scanning dependencies of target googletest
[ 1%] Creating directories for "googletest"
[ 2%] Performing download step (git clone) for "googletest"
Cloning into 'googletest-src'...
Note: checking out '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 performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at ec44c6c1 Merge pull request #821 from mazong1123/master
[ 33%] No patch step for "googletest"
[ 40%] Performing update step for "googletest"
[ 55%] No configure step for "googletest"
[ 66%] No install step for "googletest"
[ 77%] No build step for "googletest"
[ 88%] No test step for "googletest"
[100%] Completed 'googletest'
[100%] Built target googletest
-- Compiler identification is AppleClang 10.0.1.10010046
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Found PythonInterp: /usr/bin/python (found version "2.7.10")
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/build
build git:(lab0_solutions) ✘ $
```

make

type `make` to compile your project:

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!**

We will spare you the suspense. The reason for this error is that we never added `array_functions.cpp` to our `CMakeLists.txt`. Remember that **all .cpp files must be listed in the CMakeLists.txt under ADD_EXECUTABLE**

```

array_functions.h — lab-00
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

array_functions.cpp — lab-00
includes > array_functions > C array.functions.cpp > _at(int[], int, int)
2 #include "cassert"
3
4
5 void _array_init(int a[], int size, int x=0){
6

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
-- Detecting C compile features - done
-- Found PythonInterp: /usr/bin/python (found version "2.7.10")
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/build
build git:(lab0_solutions) ✘ $ make
Scanning dependencies of target gtest
[ 4%] Building CXX object googletest/googlemock/gtest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 9%] Linking CXX static library ../../lib/libgtest.a
[ 9%] Built target gtest
Scanning dependencies of target basic_test
[ 14%] Building CXX object CMakeFiles/basic_test.dir/_tests/_test_files/basic_test.cpp.o
[ 19%] Linking CXX executable bin/basic_test
Undefined symbols for architecture x86_64:
"_array_init(int*, int, int)", referenced from:
  test_append(bool) in basic_test.cpp.o
  test_init_array(bool) in basic_test.cpp.o
  test_at(bool) in basic_test.cpp.o
  _print_array(int const*, int, std::__1::basic_ostream<char, std::__1::char_traits<char> >&), referenced from:
  test_append(bool) in basic_test.cpp.o
  test_init_array(bool) in basic_test.cpp.o
  test_at(bool) in basic_test.cpp.o
  test_at(bool) in basic_test.cpp.o
"_at(int*, int, int)", referenced from:
  test_at(bool) in basic_test.cpp.o
  _stb(), referenced from:
  test_stb() in basic_test.cpp.o
"_append(int*, int&, int)", referenced from:
  test_append(bool) in basic_test.cpp.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
make[2]: *** [bin/basic_test] Error 1
make[1]: *** [CMakeFiles/basic_test.dir/all] Error 2
make: *** [all] Error 2
build git:(lab0_solutions) ✘ $ 

```

make errors, zoomed in:

Here is a closer, more readable look at the errors reported by `make`

```

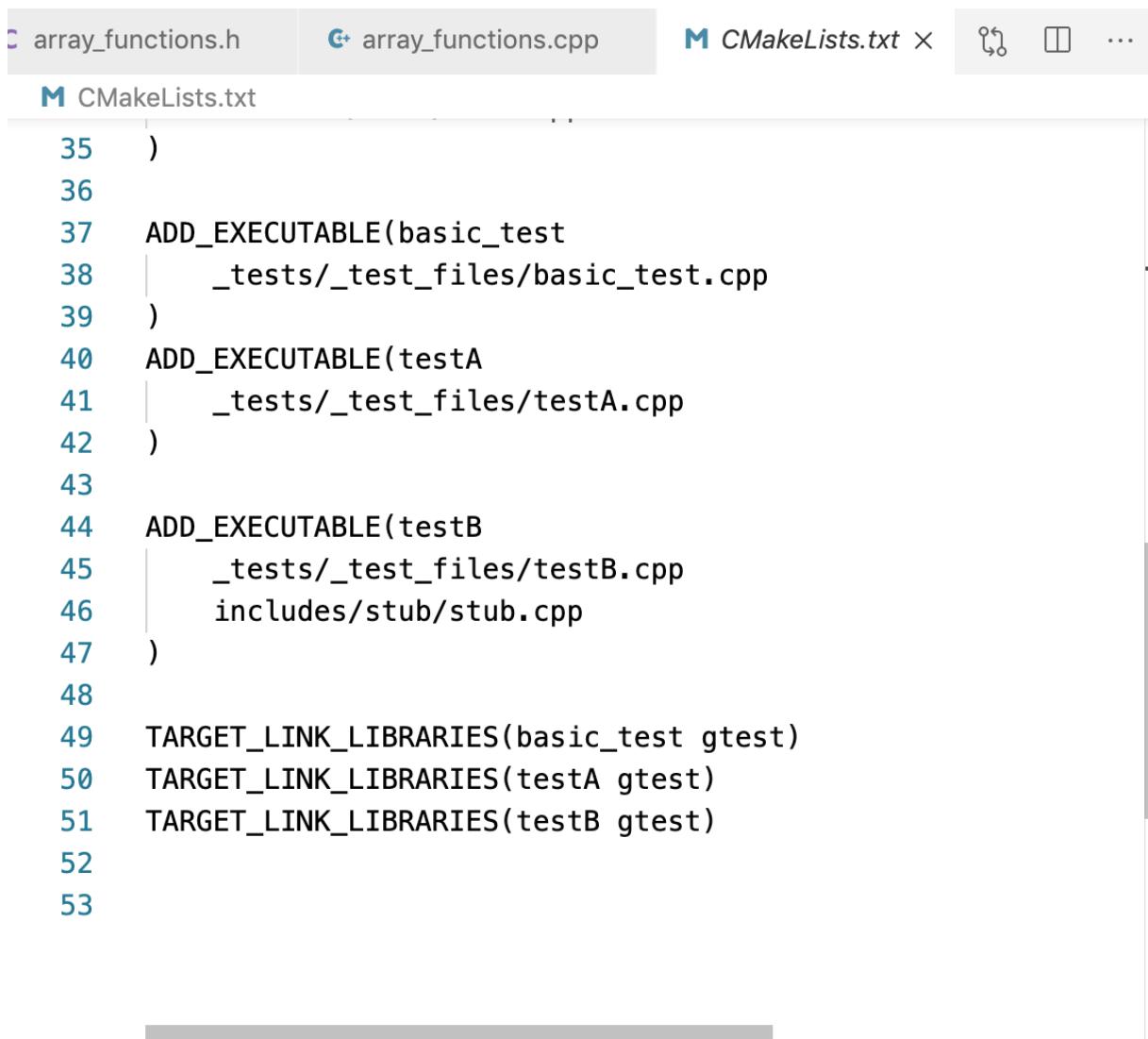
build git:(lab0_solutions) ✘ $ make
Scanning dependencies of target gtest
[ 4%] Building CXX object googletest/googlemock/gtest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 9%] Linking CXX static library ../../lib/libgtest.a
[ 9%] Built target gtest
Scanning dependencies of target basic_test
[ 14%] Building CXX object CMakeFiles/basic_test.dir/_tests/_test_files/basic_test.cpp.o
[ 19%] Linking CXX executable bin/basic_test
Undefined symbols for architecture x86_64:
"_array_init(int*, int, int)", referenced from:
  test_append(bool) in basic_test.cpp.o
  test_init_array(bool) in basic_test.cpp.o
  test_at(bool) in basic_test.cpp.o
  _print_array(int const*, int, std::__1::basic_ostream<char, std::__1::char_traits<char> >&), referenced from:
  test_append(bool) in basic_test.cpp.o
  test_init_array(bool) in basic_test.cpp.o
  test_at(bool) in basic_test.cpp.o
  test_at(bool) in basic_test.cpp.o
  test_stb() in basic_test.cpp.o
  _append(int*, int&, int), referenced from:
  test_append(bool) in basic_test.cpp.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
make[2]: *** [bin/basic_test] Error 1
make[1]: *** [CMakeFiles/basic_test.dir/all] Error 2
make: *** [all] Error 2
build git:(lab0_solutions) ✘ $ 

```

back at the `CMakeLists.txt` :

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

So, let's add it...



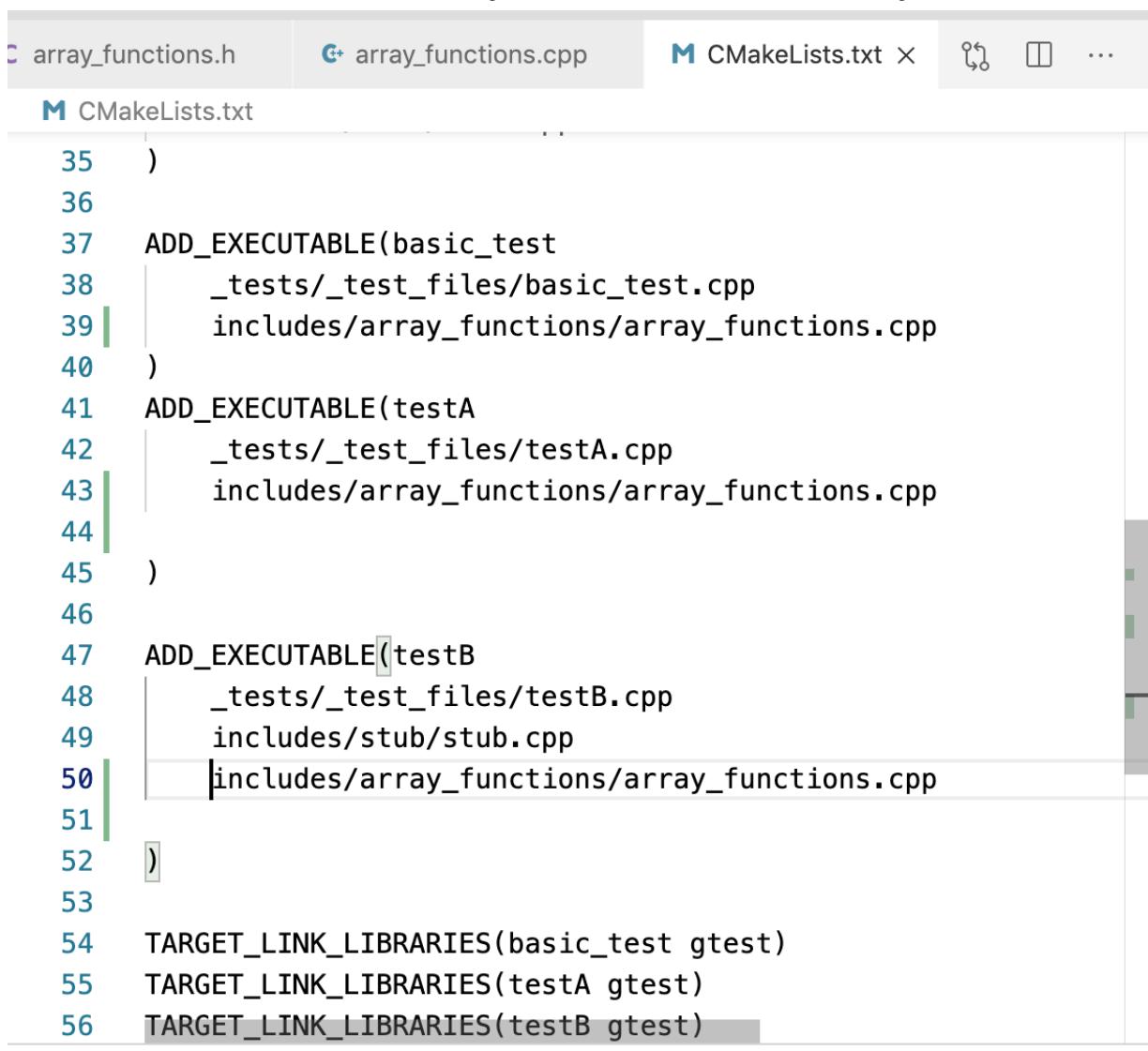
```
array_functions.h array_functions.cpp CMakeLists.txt ...  
CMakeLists.txt  
35 )  
36  
37 ADD_EXECUTABLE(basic_test  
38     _tests/_test_files/basic_test.cpp  
39 )  
40 ADD_EXECUTABLE(testA  
41     _tests/_test_files/testA.cpp  
42 )  
43  
44 ADD_EXECUTABLE(testB  
45     _tests/_test_files/testB.cpp  
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
```

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 code editor with three tabs: "array_functions.h", "array_functions.cpp", and "CMakeLists.txt". The "CMakeLists.txt" tab is active. The code contains several syntax errors, indicated by red squiggly underlines. The errors are located in lines 39, 49, and 50. The code is as follows:

```
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/array_functions/array_functions.cpp
44  )
45  )
46
47  ADD_EXECUTABLE(testB
48      _tests/_test_files/testB.cpp
49      includes/stub/stub.cpp
50      includes/array_functions/array_functions.cpp
51  )
52  )
53
54  TARGET_LINK_LIBRARIES(basic_test gtest)
55  TARGET_LINK_LIBRARIES(testA gtest)
56  TARGET_LINK_LIBRARIES(testB gtest)
```

make again:

Let's run `make` 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.

```
build git:(lab0_solutions) ✘ ☺$ make
[ 8%] Built target gtest
Scanning dependencies of target basic_test
[ 12%] Building CXX object CMakeFiles/basic_test.dir/includes/array_functions/array_functions.cpp.o
/Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/includes/array_functions/array_functions.cpp:5:41: error: redefinition of default argument
void _array_init(int a[], int size, int x=0){
^~~~~~
/Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/includes/array_functions/array_functions.h:7:41: note: previous definition is here
void _array_init(int a[], int size, int x=0);
^~~~~~
/Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/includes/array_functions/array_functions.cpp:25:57: error: redefinition of default argument
ostream& _print_array(const int a[], int size, ostream& outs = cout){
^~~~~~
/Users/sassanbarkeshli/Downloads/_sandboxes/cs3a-projects/lab-00/includes/array_functions/array_functions.h:15:57: note: previous definition is here
ostream& _print_array(const int a[], int size, ostream& outs = cout);
^~~~~~
2 errors generated.
make[2]: *** [CMakeFiles/basic_test.dir/includes/array_functions/array_functions.cpp.o] Error 1
make[1]: *** [CMakeFiles/basic_test.dir/all] Error 2
make: *** [all] Error 2
build git:(lab0_solutions) ✘ ☺$ █
```

Fix the _print_array function...

```
24
25     ostream& _print_array(const int a[], int size, ostream& outs = cout){
26
27         return outs;
28     }
29 }
```

by removing the default value = cout

```
array_functions.h      array_functions.cpp ×      CMakeLists.txt
includes > array_functions > array_functions.cpp > _print_array(const int [], int, ostream &
17 }
18
19 int& _at(int a[], int size, int pos){
20
21     return a[0];
22 }
23
24
25 ostream& _print_array(const int a[], int size, ostream& outs){
26
27     return outs;
28 }
29 }
```

Same with _array_init :

```
4
5 void _array_init(int a[], int size, int x=0){
6
7
8 }
```

```
4
5 void _array_init(int a[], int size, int x){
6
7
8 }
9
```

make 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 the `basic_test` from the `bin/` directory. 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 VS Code interface with two code editors and a terminal window.

- Left Sidebar:** Shows the project structure under "EXPLORER". It includes files like basic_test.cpp, testA.cpp, testB.cpp, array_functions.h, stub.cpp, stub.h, .gitignore, CMakeLists.txt, main.cpp, and README.md.
- Code Editors:**
 - Top Editor: basic_test.cpp. Content includes a TEST macro definition and a basic_test function.
 - Bottom Editor: array_functions.cpp. Content includes an _array_init function.
- Terminal:** Shows the build process of a CMake project named "lab0_solutions". It includes commands like "make", "ctest", and "./bin/basic_test". The output also shows the execution of the test case.
- Bottom Status Bar:** Displays file paths, build status, and other system information.

**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.

```
[100%] Built target gtest_main
build git:(lab0_solutions) ✘ 🐸$ cd ..
lab-00 git:(lab0_solutions) ✘ 🐸$ git status
On branch lab0_solutions
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   _tests/_test_files/basic_test.cpp
    new file:   includes/array_functions/array_functions.cpp
    new file:   includes/array_functions/array_functions.h

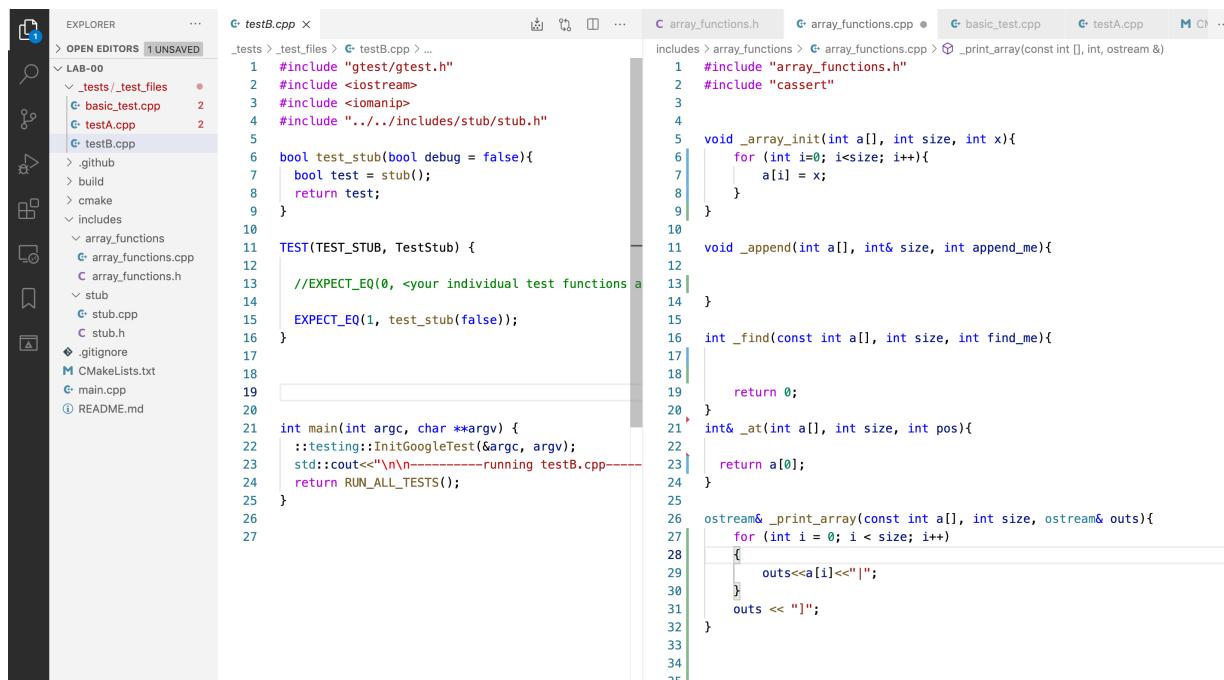
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CMakeLists.txt
    modified:   _tests/_test_files/basic_test.cpp
    modified:   _tests/_test_files/testA.cpp
    modified:   includes/array_functions/array_functions.cpp

lab-00 git:(lab0_solutions) ✘ 🐸$ git add .
lab-00 git:(lab0_solutions) ✘ 🐸$ git commit -m "success on make with stubs"
[lab0_solutions 4c5b80c] success on make with stubs
  5 files changed, 286 insertions(+), 12 deletions(-)
  create mode 100644 includes/array_functions/array_functions.cpp
  create mode 100644 includes/array_functions/array_functions.h
lab-00 git:(lab0_solutions) ✘ 🐸$
```

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.



```
EXPLORER OPEN EDITORS 1 UNSAVED
LAB-00
  _tests/_test_files
    basic_test.cpp
    testA.cpp
  testB.cpp
  .github
  build
  cmake
  includes
    array_functions
      array_functions.cpp
      array_functions.h
    stub
      stub.cpp
      stub.h
  .gitignore
  CMakeLists.txt
  main.cpp
  README.md

C testB.cpp
#include "gtest/gtest.h"
#include <iostream>
#include <iomanip>
#include "../../../../includes/stub/stub.h"

bool test_stub(bool debug = false){
    bool test = stub();
    return test;
}

TEST(TEST_STUB, TestStub) {
    //EXPECT_EQ(0, <your individual test functions a>)
    EXPECT_EQ(1, test_stub(false));
}

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

array_functions.h
#include "array_functions.h"
#include <cassert>

void _array_init(int a[], int size, int x){
    for (int i=0; i<size; i++){
        a[i] = x;
    }
}

void _append(int a[], int& size, int append_me){
    if (size == 0) {
        a[0] = append_me;
        size++;
    } else {
        int temp = a[size-1];
        a[size-1] = append_me;
        a[size] = temp;
        size++;
    }
}

int _find(const int a[], int size, int find_me){
    for (int i=0; i<size; i++) {
        if (a[i] == find_me) {
            return i;
        }
    }
    return -1;
}

ostream& _print_array(const int a[], int size, ostream& outs){
    for (int i = 0; i < size; i++) {
        outs<<a[i]<<"|";
    }
    outs << "]";
}
```

■ 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.

```

1  #include "array_functions.h"
2  #include "cassert"
3
4
5  void _array_init(int a[], int size, int x){
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 int _at(int a[], int size, int pos){
21
22     return a[0];
23 }
24
25 ostream& _print_array(const int a[], int size, ostream& outs){
26
27     for (int i=0; i<size; i++){
28
29         outs << a[i] << "|";
30
31     }
32
33     outs << "]";
34
35 }
36
37
38
39
40 > TEST(TEST_STUB, TestStub) {
41
42     TEST(TEST_ARRAY, TestInit) {
43
44
45 //EXPECT_EQ(1, <your individual test functions are called here>)
46
47     EXPECT_EQ(1, test_init_array(false));
48
49 }
50
51
52 }
53

```

#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.

```

10  #include "array_functions.h"
11  #include "cassert"
12
13  void _array_init(int a[], int size, int x){
14      for (int i = 0; i < size; i++)
15          a[i] = x;
16  }
17  void _append(int a[], int& size, int append_me){
18      int size = 5;
19      const char tabs[] = "\t-----\t";
20      if (append_me)
21          cout << tabs << "size: " << size << ", init array to -1" << endl;
22      _array_init(a, size, -1);
23      if(debug){
24          cout << tabs;
25          _print_array(a, size);
26          cout << endl;
27      }
28      for (int i = 0; i < size; i++){
29          if (a[i] != -1){
30              cout << "FAILED: Expected -1 at a[" << i
31              << "] but found: " << a[i] << endl;
32              return false;
33          }
34      }
35
36      return true;
37  }
38
39  TEST(TEST_STUB, TestStub) {
40      TEST(TEST_ARRAY, TestInit) {
41          //EXPECT_EQ(1, <your individual test functions are called here>)
42          EXPECT_EQ(1, test_init_array(false));
43      }
44  }
45
46  EXPECT_EQ(1, test_init_array(false));
47
48  EXPECT_EQ(1, test_init_array(true));
49
50  EXPECT_EQ(1, test_init_array(true));
51
52 }

```

make and RUN!!

This time, we will run `make` successfully and then, we run the `testB` executable by typing `./bin/testB`

This means execute the file named `testB` that is located in the `bin` folder which is under the `current folder`. The `bin` folder is created by `make`

remember that `.` means current folder and is not optional. You **must** include the dot in the call to execute `testB`

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 VS Code interface with several tabs open:

- EXPLORER**: Shows the project structure with files like `basic_test.cpp`, `testA.cpp`, `testB.cpp`, `array_functions.cpp`, `array_functions.h`, `stub.cpp`, and `stub.h`.
- testB.cpp**: The active tab, containing code for a test function.
- array_functions.cpp**: Another tab with code for array manipulation.
- basic_test.cpp**: A third tab.
- TERMINAL**: Shows the build process for `testB`:


```
build git:(lab0_solutions) ✘ $> make
[ 8%] Built target gtest
[ 20%] Built target basic_test
[ 33%] Built target testA
Scanning dependencies of target testB
[ 37%] Building CXX object CMakeFiles/testB.dir/_tests/_test_files/testB.cpp.o
[ 41%] Linking CXX executable bin/testB
[ 50%] Built target testB
[ 62%] Built target main
[ 75%] Built target gmock
[ 91%] Built target gmock_main
[100%] Built target gtest_main
build git:(lab0_solutions) ✘ $> ./bin/testB
-----running testB.cpp-----
[=====] Running 2 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)

[=====] 1 test from TEST_ARRAY
[ RUN ] TEST_ARRAY.TestInit
[ OK ] TEST_ARRAY.TestInit (0 ms)
[=====] 1 test from TEST_ARRAY (0 ms total)

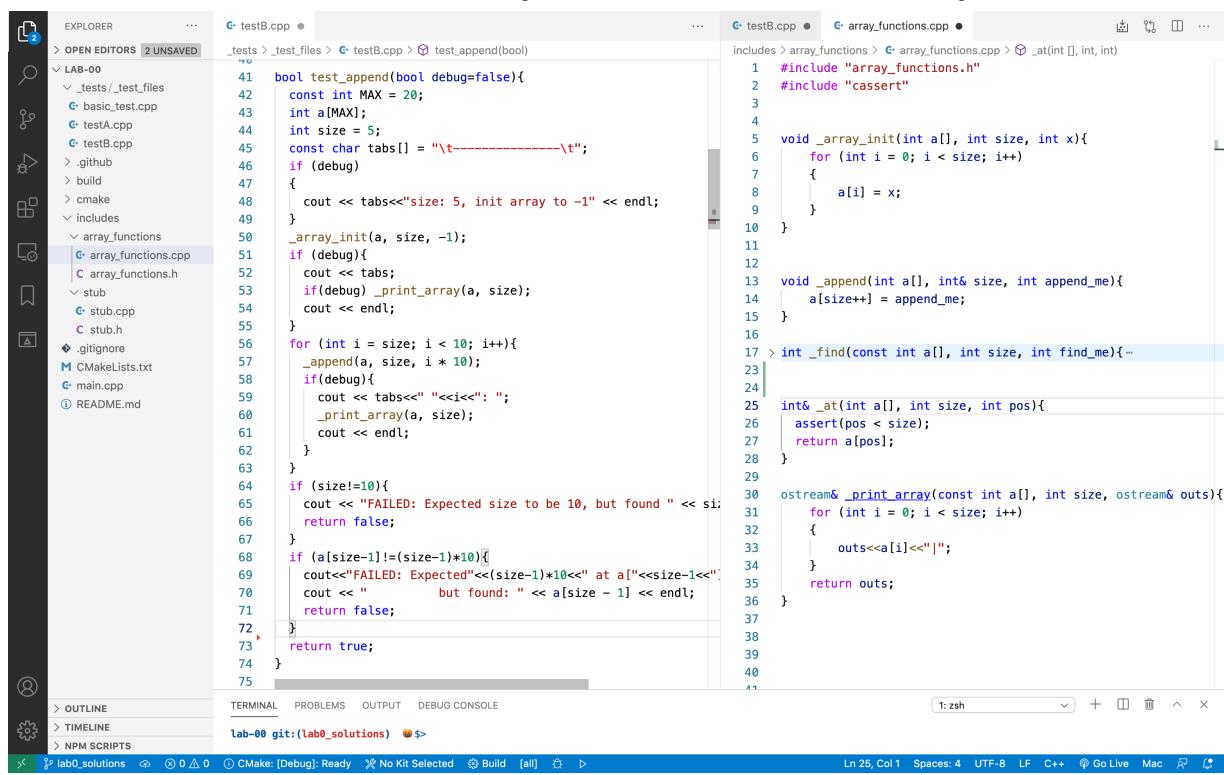
[=====] Global test environment tear-down
[=====] 2 tests from 2 test cases ran. (0 ms total)
[ PASSED ] 2 tests.
build git:(lab0_solutions) ✘ $>
```

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.

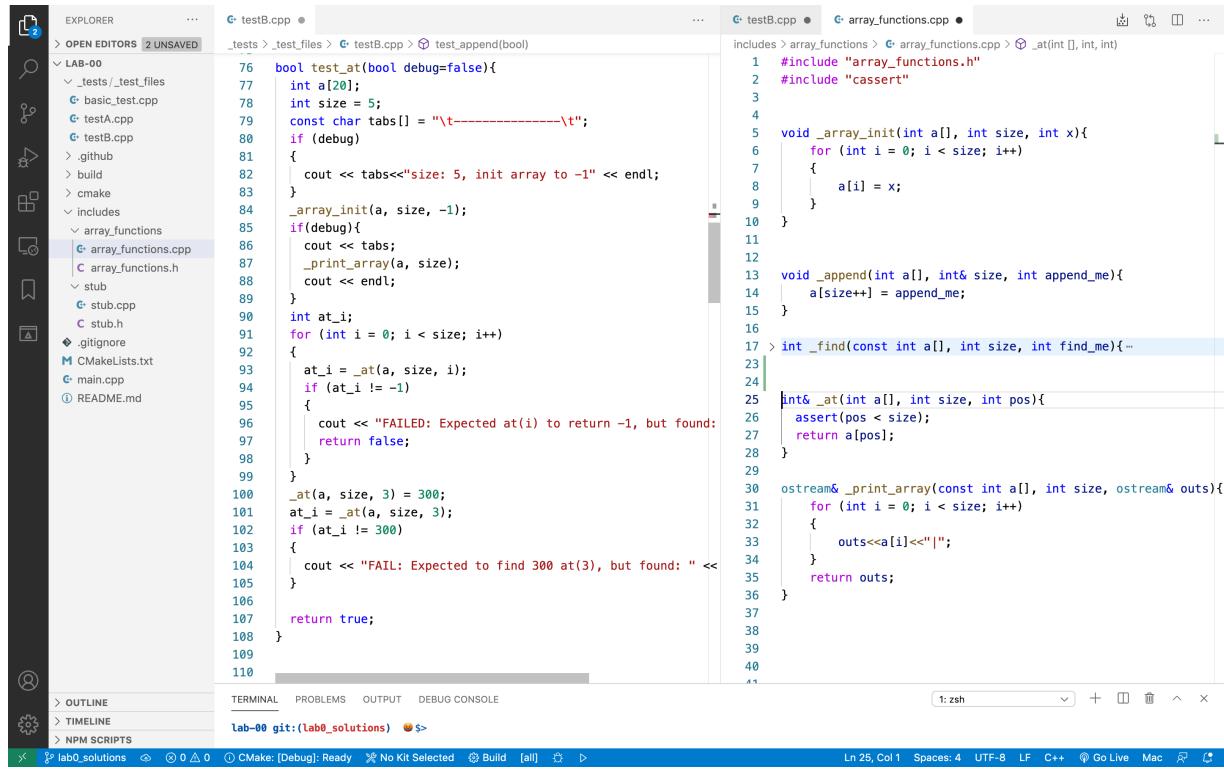


```

41     bool test_append(bool debug=false){
42         const int MAX = 20;
43         int a[MAX];
44         int size = 5;
45         const char tabs[] = "\t-----\t";
46         if (debug)
47         {
48             cout << tabs<<"size: 5, init array to -1" << endl;
49         }
50         _array_init(a, size, -1);
51         if (debug){
52             cout << tabs;
53             if(debug) _print_array(a, size);
54             cout << endl;
55         }
56         for (int i = size; i < 10; i++){
57             _append(a, size, i * 10);
58             if(debug){
59                 cout << tabs<<" " << i << ": ";
60                 _print_array(a, size);
61                 cout << endl;
62             }
63         }
64         if (size!=10){
65             cout << "FAILED: Expected size to be 10, but found " << size;
66             return false;
67         }
68         if (a[size-1]!=(size-1)*10){
69             cout<<"FAILED: Expected "<<(size-1)*10<<" at a["<<size-1<<".
70             cout << " but found: " << a[size - 1] << endl;
71             return false;
72         }
73     return true;
74 }
75 
```

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.



```

76     bool test_at(bool debug=false){
77         int a[20];
78         int size = 5;
79         const char tabs[] = "\t-----\t";
80         if (debug)
81         {
82             cout << tabs<<"size: 5, init array to -1" << endl;
83         }
84         _array_init(a, size, -1);
85         if(debug){
86             cout << tabs;
87             _print_array(a, size);
88             cout << endl;
89         }
90         int at_i;
91         for (int i = 0; i < size; i++)
92         {
93             at_i = _at(a, size, i);
94             if (at_i != -1)
95             {
96                 cout << "FAILED: Expected at(i) to return -1, but found: " << at_i;
97                 return false;
98             }
99         }
100        _at(a, size, 3) = 300;
101        at_i = _at(a, size, 3);
102        if (at_i != 300)
103        {
104            cout << "FAIL: Expected to find 300 at(3), but found: " << at_i;
105        }
106    return true;
107 }
108 
```

make and run testB.cpp again:

Let's make and run testB to make sure our `test_append` and `test_at` pass:

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows the project structure under `LAB-00`, including files like `basic_test.cpp`, `testA.cpp`, `testB.cpp`, `array_functions.h`, and `array_functions.cpp`.
- Code Editors:** Four tabs are open:
 - `array_functions.cpp`: Contains implementation for `_array_init`, `_append`, `_find`, and `_at`.
 - `basic_test.cpp`: Contains test cases for `TEST_ARRAY`.
 - `testB.cpp`: Contains the main test function.
 - `testA.cpp`: Contains another test function.
- Terminal:** Displays the output of a CTest run:


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

[=====] 3 tests from TEST_ARRAY
[RUN] [OK] TEST_ARRAY.TestInit (0 ms)
[RUN] [OK] TEST_ARRAY.TestAppend (0 ms)
[RUN] [OK] TEST_ARRAY.TestAppend (0 ms)
[RUN] [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. (0 ms total)
[PASSED] 4 tests
```

■ 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 `make` and run `testB` once again.

Once you have successfully run `testB` with `_find`, you `git add` and `commit` your changes:

```
build git:(lab0_solutions) ✘ 🎉$> git status
On branch lab0_solutions
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   ../_tests/_test_files/basic_test.cpp
    modified:   ../_tests/_test_files/testB.cpp

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

    modified:   ../_tests/_test_files/basic_test.cpp
    modified:   ../_tests/_test_files/testB.cpp
    modified:   ../includes/array_functions/array_functions.cpp

build git:(lab0_solutions) ✘ 🎉$> git add .
build git:(lab0_solutions) ✘ 🎉$> git commit -m "PASSED: testB passed all tests."
[lab0_solutions 11c5bfd] PASSED: testB passed all tests.
  2 files changed, 73 insertions(+), 130 deletions(-)
  rewrite _tests/_test_files/basic_test.cpp (69%)
  copy _tests/_test_files/{basic_test.cpp => testB.cpp} (89%)
```

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 VS Code interface with the following details:

- EXPLORER:** Shows the project structure with files like `_tests/_test_files/basic_test.cpp`, `testB.cpp`, `testA.cpp`, `array_functions.h`, etc.
- CODE EDITOR:** Displays four tabs: `array_functions.cpp`, `basic_test.cpp`, `testB.cpp`, and `array_functions.h`. The `array_functions.h` tab contains header-only code for arrays.
- TERMINAL:** Shows the output of running `basic_test.cpp`. It includes the command `g++ basic_test.cpp -o basic_test`, the test environment setup, and the execution of the test cases. The output shows the state of an array and the results of the search operation.
- STATUS BAR:** Shows the current file is `lab0_solutions`, build status as `Ready`, and other system information like Ln 53, Col 35, Spaces: 2, etc.

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`

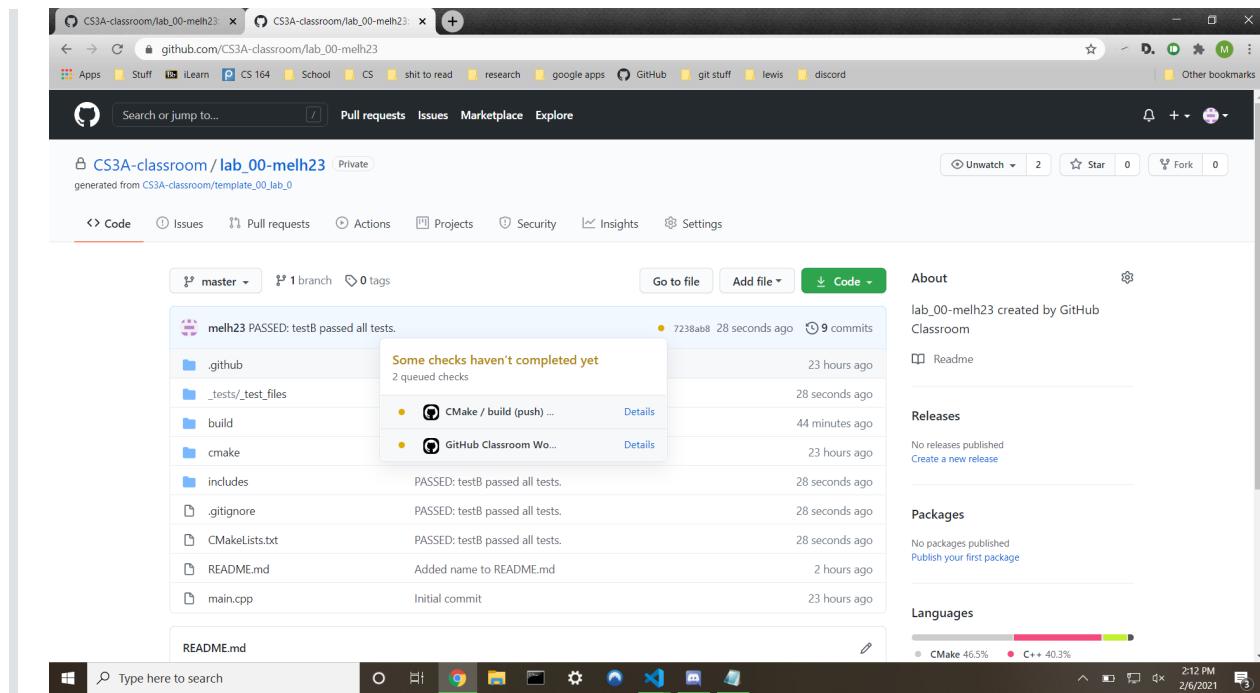
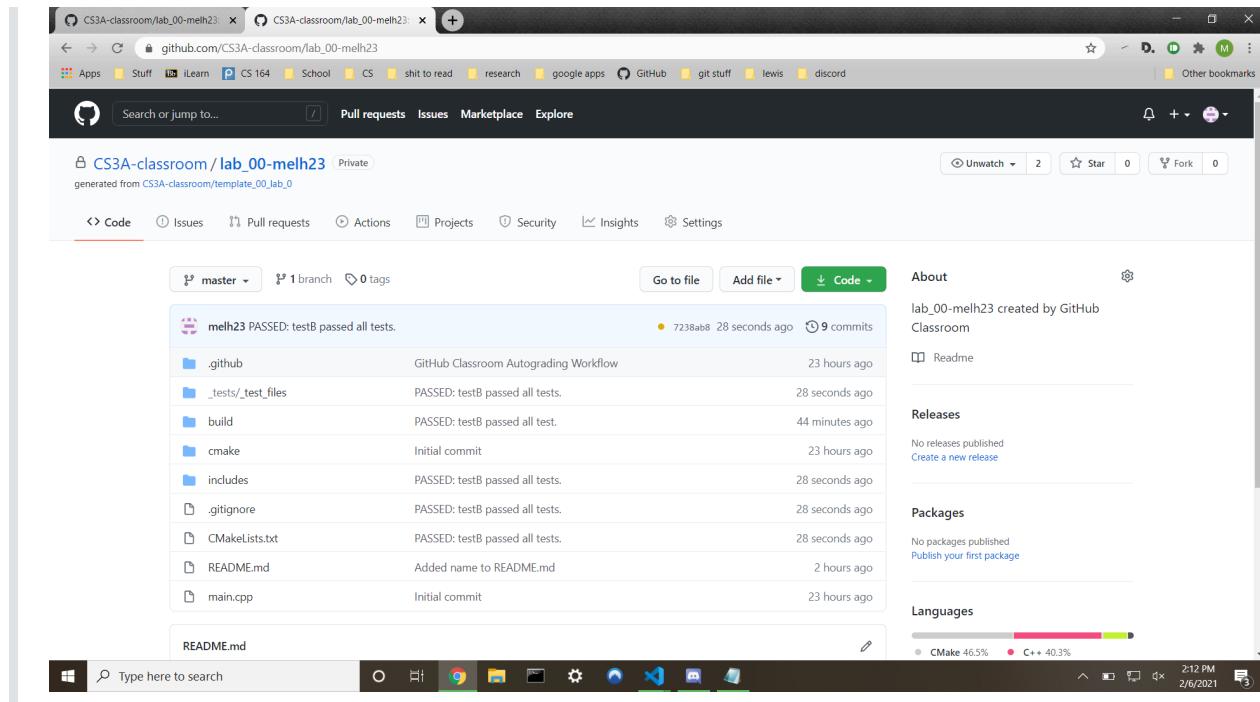
```
build git:(lab0_solutions) ✘ 🎉$> git status
On branch lab0_solutions
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   ../../_tests/_test_files/basic_test.cpp
    modified:   ../../_tests/_test_files/testB.cpp
    modified:   ../../includes/array_functions/array_functions.cpp

no changes added to commit (use "git add" and/or "git commit -a")
build git:(lab0_solutions) ✘ 🎉$> cd ..
lab-00 git:(lab0_solutions) ✘ 🎉$> git add .
lab-00 git:(lab0_solutions) ✘ 🎉$> git commit -m "PASSED: testB, basic_test"
[lab0_solutions 765dc96] PASSED: testB, basic_test
  3 files changed, 181 insertions(+), 45 deletions(-)
```

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.



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 has 1 branch and 0 tags. A green checkmark indicates that 'melh23 PASSED: testB passed all tests.' The 'Actions' tab is selected, showing a table of recent actions:

Action	Description	Time Ago
All checks have passed	2 successful checks	23 hours ago
CMake / build (push)	Details	1 minute ago
GitHub Classroom Wo...	Details	1 hour ago
		23 hours ago
	PASSED: testB passed all tests.	1 minute ago
	PASSED: testB passed all tests.	1 minute ago
	PASSED: testB passed all tests.	1 minute ago
	Added name to README.md	2 hours ago
	Initial commit	23 hours ago

On the right side, there are sections for 'About', 'Readme', 'Releases', 'Packages', and 'Languages'. The 'Languages' section shows C++ at 62.7% and CMake at 29.3%. The bottom of the screen shows a Windows taskbar with various icons.

It will take you to this page:

The screenshot shows the 'Actions' logs for the 'CS3A-classroom/lab_00-melh23/runs/1846556585' run. The 'Autograding' section is expanded, showing the log output for the 'Run education/autograding@v1' step:

```

-----running basic_test.cpp-----
[=====] 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|80|]
[-1|-1|-1|-1|-1|50|60|70|80|80|90|]
found 70 at: 70
changing 70 to:
[-1|-1|-1|-1|-1|50|60|70|80|80|]

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

```

A screenshot of a Windows desktop environment. At the top, there's a taskbar with several pinned icons: Apps, Stuff, iLearn, CS 164, School, CS, shit to read, research, google apps, GitHub, git stuff, lewis, and discord. Below the taskbar is a browser window displaying a GitHub pull request. The pull request has passed all tests, as indicated by the green checkmark icon and the message "PASSED: testB passed all tests." The "Actions" tab is selected, showing a GitHub Classroom Workflow run. The "Autograding" section shows a log for a run named "Run education/autograding@v1". The log output is as follows:

```
1/8 | 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
188
189 :1e41c4b3-f766-4da4-999e-293106bf9ee5:
190
191 All tests passed
192
193 ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨ ✨
194
195 Points 50/50
196
```

The log concludes with a summary of 50/50 points. Below the log, there are two more collapsed sections: "Post Run actions/checkout@v2" and "Complete job". The bottom right corner of the screen shows the date and time as 2/6/2021 2:20 PM.

CS3A-classroom / lab_00-melh23 Private

generated from [CS3A-classroom/template_00_lab_0](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [...](#)

[master](#) [Go to file](#) [Add file](#) [Code](#)

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

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

File List

- melh23 added .bat files** 6 minutes ago (4 commits)
- .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

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's a GitHub Classroom Workflow section with a status of 'on: push' and 2 runs. The 'Autograding' workflow is highlighted in blue, showing 1 run with steps: 'Set up job' (3s) and 'Run actions/checkout@v2' (1s). 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. 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. Below the header, the repository name and a green checkmark icon indicating a 'PASSED' status are displayed. The commit hash shown is 7238abb8.

The repository has two workflows: 'GitHub Classroom Workflow' (on: push) and 'Autograding' (on: push). The 'Autograding' workflow is expanded, showing its logs. The logs detail the execution of 'Run education/autograding@v1'. The output shows the execution of 'testB.cpp' and 'testB'. The logs indicate that all tests passed, with a total of 50/50 points. The logs also show the use of CMake and the GitHub Classroom environment.

```
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
```

