

Question 28: Will be asked by the instructor to check if you have followed the hints for me.

Instructions : Complete all steps below to checkout starter code from GitHub, update files and CMake scripts, configure and test your code, and confirm your score with GitHub actions.

Before getting started, you will need a GitHub account and a way to edit repositories. If you do not already have a GitHub authentication token or SSH key setup, a step-by-step guide is available [here](#).

If at any step you run into issues or get confused, scroll to the end of this document for hints and common issues.

PART 1: GETTING STARTED WITH GITHUB

1. **Click on your GitHub Classroom Link, provided to you by your instructor.**
2. **Copy your repository address.** Click the green 'Code' button to find this link. **Make sure the 'HTTPS' button is selected if you are using authentication tokens.**
3. **Clone the repository.** On your local machine, open up a terminal/Command Line. Clone the repository with the command 'git clone' followed by the repository address copied in the previous step. **If you are asked for a password, use the GitHub authentication token (instructions for this token are linked at the top of this PDF).**
4. **Move into the folder you just checked out.** You should be able to see the folder you just checkout out with the command 'ls'. You can enter this folder with 'cd foldername', where foldername is the name of your new folder.
5. **Pull submodule updates.** Initial code, configuration, and testing environments are provided for you in a separate GitHub repository. You cannot change this code. Any updates to your homework, including updated tests, will be added in this submodule. Make sure to pull the latest changes to this submodule with the following:
 - (a) 'git submodule init'
 - (b) 'git submodule update --remote'

PART 2: CMAKE

CMake is a group of tools for compiling code on any given computer. The library that you cloned in Part 1 must be compiled with CMake. Follow the steps below to complete basic tasks with CMake.

1. **Create a build folder.** All CMake commands should be run within a build folder so that they can be easily cleaned up. Create a new folder in the homework0 repository called 'build'
2. **Change directory into build.** Change the current directory to build (e.g. 'cd build')
3. **Configure the current CMake directory.** To configure your compilation, type 'cmake ..' from within the build directory
4. **Compile the codebase.** After the configuration successfully completes, you can compile the library with the command 'make'. You should see an error that the linker cannot find the function 'return0()'
5. **Create a C or C++ file within the main repository directory.** Leave the build directory, stepping back one folder ('cd ..') and create a new file.

6. **Edit this File.** Include the header file 'src.hpp' and create a new method called 'return0' which returns the integer 0. Note there is no 'main' method, this is a method within a library, and it has already been declared within the header file (located in the submodule folder).
7. **Edit CMakeLists.txt.** Open the file 'CMakeLists.txt' from the main directory. A library called 'homework' is created near the bottom of this file. Add your file to this library.
8. **Reconfigure and compile the library.** Go back into the build directory and run the configuration and compile commands again (from steps 3 and 4 above).

PART 3: GOOGLETEST

Googletest is a testing environment, which will test your code for correctness. These tests are provided for you, within the 'tests' directory of the submodule folder.

1. **Open the file 'unit_tests.cpp'.** This file is within the 'tests' folder of the provided submodule. This is a simple test written with googletest. Make sure you can understand what is being tested in this file.
2. **Test for Correctness.** To test that your code changes and googletest unit tests are working, go back to the build folder and recompile your code ('make'). Then, run the tests with 'make test'. This will run my provided unit tests. You should see tests pass.
3. **Update Compile Script.** Add all configuration and compilation commands to the file 'compile.sh'. This is the file that will GitHub Classroom will us to compile your code.
4. **Retest Correctness.** Delete your build folder and instead run only the following commands from the main repository folder:
 - (a) 'sh compile.sh'
 - (b) 'cd build'
 - (c) 'make test'

You should again see that you pass the unit tests.

PART 4: COMMITTING AND CHECKING

Once all of your tests are passing, you need to push the code back to GitHub. **For larger tasks, you should do this step even before all tasks are complete as a way to checkpoint your code.**

1. **Pull Repository Updates.** The command 'git pull' will pull any updates from your repository. If your local code is not up-to-date with the online repository, you will not be able to push new changes without first pulling.
2. **Check Repository Status.** The command 'git status' will show which files have been changed, added, or deleted since your last pulled code.
3. **Add Files.** Any files that have been changed or added to your repository can be added with any of the following commands. Note, this stages these changes, but they will still need to be committed and pushed before they appear in your online repository.
 - (a) 'git add *filename*': will add the file named *filename* to be committed. Replace *filename* with the name of the file that you want to add.
 - (b) 'git add *': will add all files in your current directory.
 - (c) 'git add -u': will add only files that have previously been added to the repository and have since been updated.
4. **Commit Changes.** After adding updates and new files, you need to commit these changes with 'git commit -m *message*' where *message* describes your update.

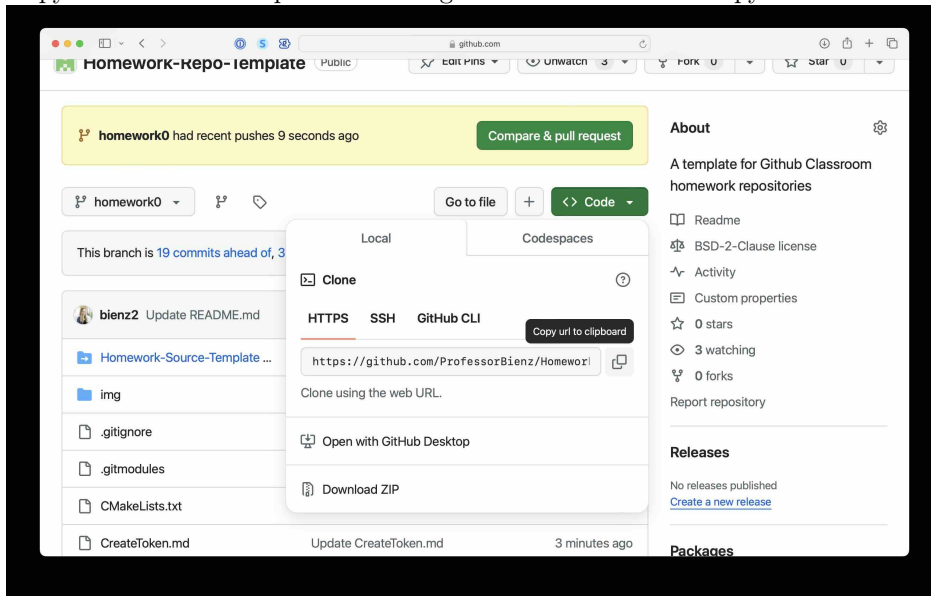
5. **Push Changes.** After an update has been committed and tagged with a message, push your changes to your online repository with 'git push'. If asked for a password, you will need to enter your GitHub authentication token again.
6. **Check Repository.** Go back to your GitHub repository and refresh the page if needed. You should see your updated code in your repository.
7. **Check GitHub Actions.** Click on the button 'Actions' and find the message you used with your most recent commit. If there is a yellow circle next to your commit, the workflow is still in progress and you will have to wait until it finishes to see if tests pass. If the circle is red, at least one test has failed. If the circle is green, all tests have passed!
8. **Analyze GitHub Actions.** Click on the action, regardless of the circle color. Note:
 - (a) The steps Set Up job and Checkout code should work. If either of these has an error, you may need to reach out for additional help debugging it.
 - (b) Click on all additional headers. **They may have a white checkmark even if code is not compiled correctly!**

Hints and Common Issues

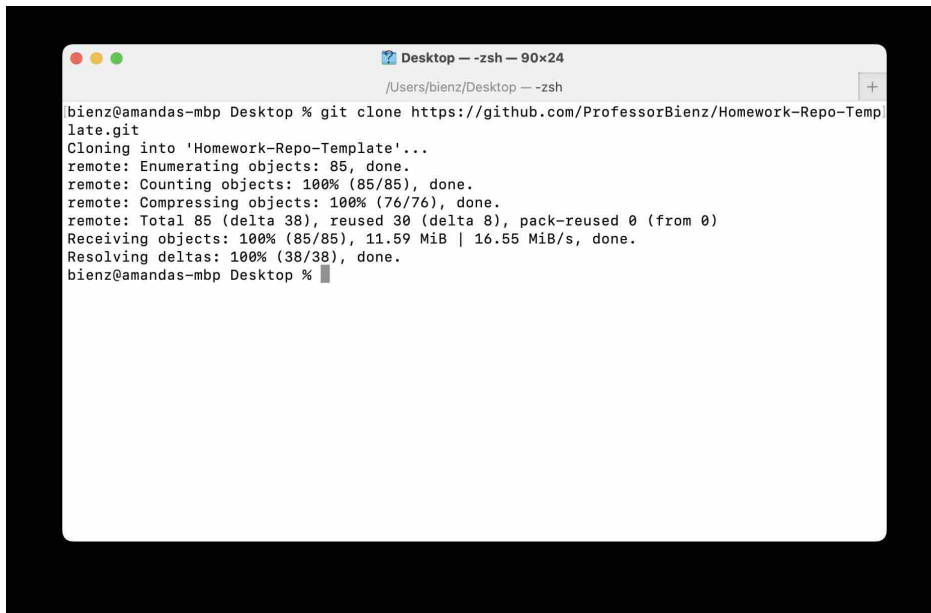
Included below are hints, including screenshots accomplishing many of the tasks listed throughout this document. At the end of this section, there is also a list of common issues and strategies for solving them.

Hints

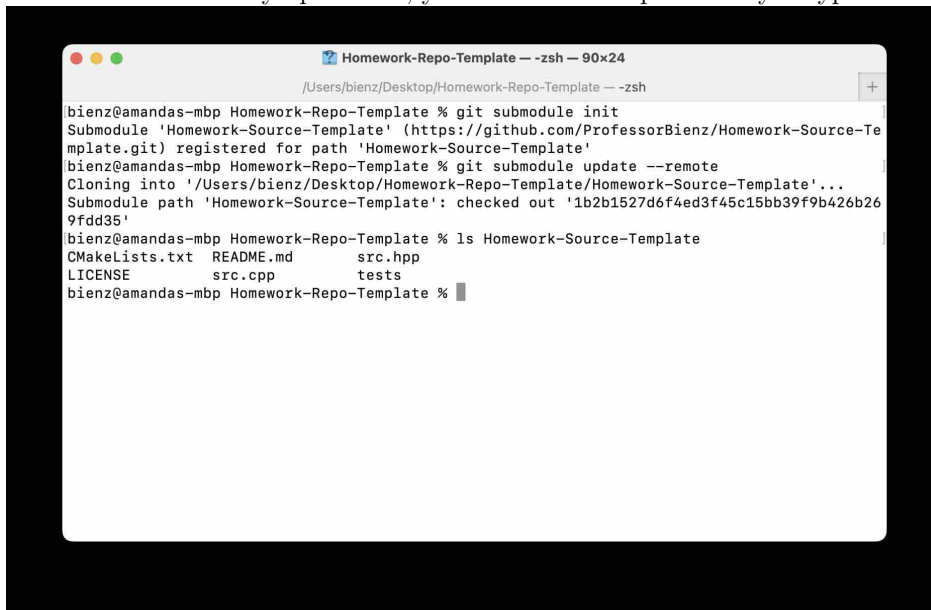
- (1.2:) The image below shows how to find your repository address. This is needed to clone your repository. Copy this address and paste it after 'git clone' for an initial copy of the code.



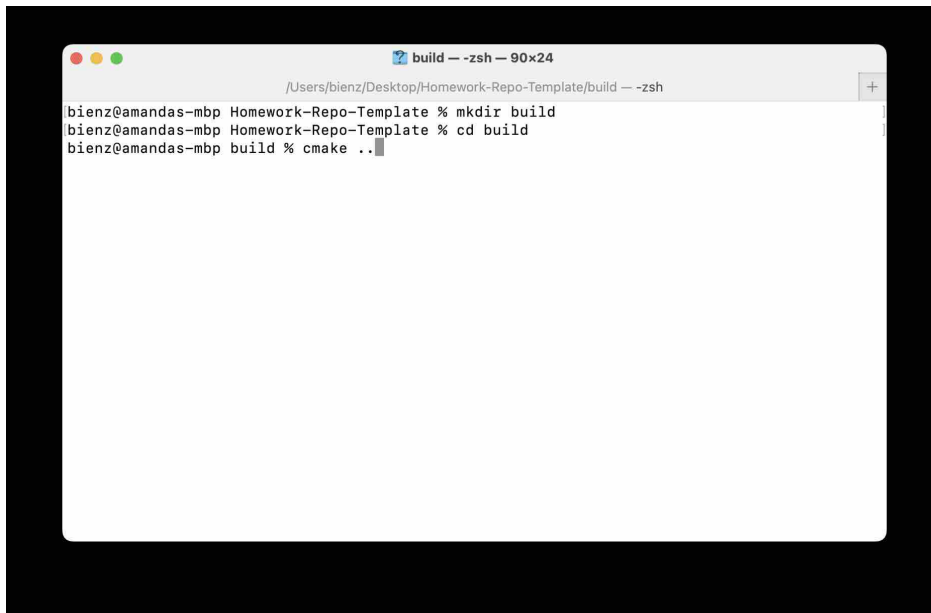
- (1.3:) After you clone your repository, you should see similar output. If you get an error, check your that you have correctly copied the HTTPS address from your repository. If you are asked for a password, you need to use your GitHub authentication token, not your main GitHub password.

A terminal window titled 'Desktop -- zsh -- 90x24' with the path '/Users/bienz/Desktop -- zsh'. The user 'bienz@amandas-mbp' runs the command 'git clone https://github.com/ProfessorBienz/Homework-Repo-Template.git'. The output shows the cloning process: 'Cloning into 'Homework-Repo-Template'...', 'remote: Enumerating objects: 85, done.', 'remote: Counting objects: 100% (85/85), done.', 'remote: Compressing objects: 100% (76/76), done.', 'remote: Total 85 (delta 38), reused 30 (delta 8), pack-reused 0 (from 0)', 'Receiving objects: 100% (85/85), 11.59 MiB | 16.55 MiB/s, done.', 'Resolving deltas: 100% (38/38), done.', and the prompt returns to 'bienz@amandas-mbp Desktop %'.

- (1.5) After you update your submodule, you may see similar output to the image below. However, if your submodule was already up-to-date, you will see no output when you type this command.

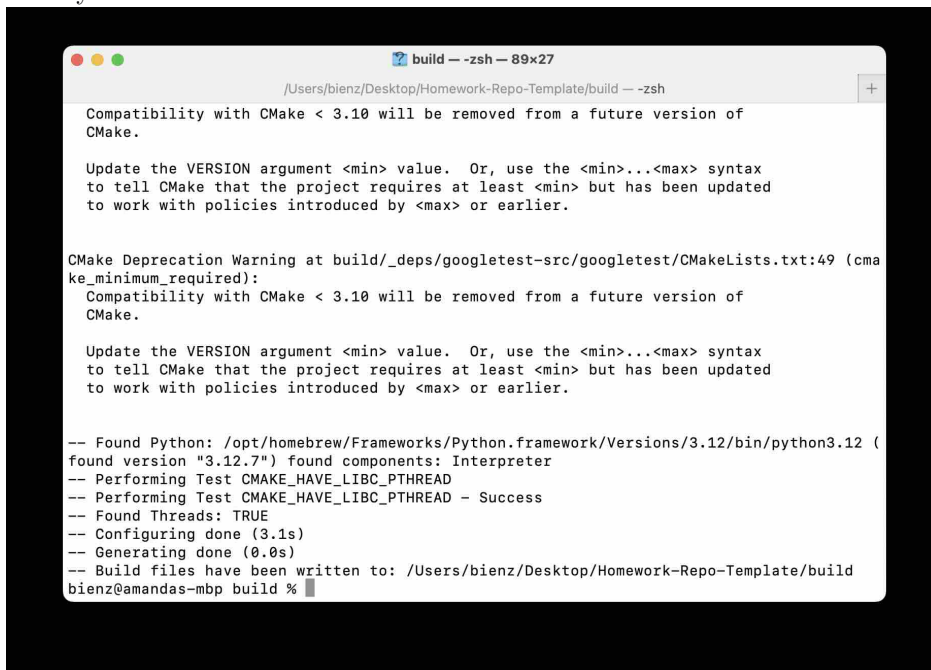
A terminal window titled 'Homework-Repo-Template -- zsh -- 90x24' with the path '/Users/bienz/Desktop/Homework-Repo-Template -- zsh'. The user 'bienz@amandas-mbp' runs 'git submodule init', which outputs 'Submodule 'Homework-Source-Template' (https://github.com/ProfessorBienz/Homework-Source-Template.git) registered for path 'Homework-Source-Template''. Then, they run 'git submodule update --remote', which outputs 'Cloning into '/Users/bienz/Desktop/Homework-Repo-Template/Homework-Source-Template'...', 'Submodule path 'Homework-Source-Template': checked out '1b2b1527d6f4ed3f45c15bb39f9b426b269fdd35'', and then 'ls Homework-Source-Template', which outputs a directory listing: 'CMakeLists.txt README.md src.hpp', 'LICENSE src.cpp tests'. The prompt returns to 'bienz@amandas-mbp Homework-Repo-Template %'.

- (2.3) The following image shows how to create a new build folder, move into that folder, and run the 'cmake' configuration script from within that folder. **Make sure you have two dots after cmake, pointing to the CMakeLists.txt file, which is in the parent directory of your current location (the build folder).**

A terminal window titled 'build -- zsh -- 90x24' with the path '/Users/bienz/Desktop/Homework-Repo-Template/build -- zsh'. The prompt is 'bienz@amandas-mbp Homework-Repo-Template %'. The user enters 'mkdir build', then 'cd build', and finally 'cmake ..'.

```
bienz@amandas-mbp Homework-Repo-Template % mkdir build
bienz@amandas-mbp Homework-Repo-Template % cd build
bienz@amandas-mbp build % cmake ..
```

CMake will configure your code, and you should see the line saying ‘Build files have been written to...’, pointing to a location on your computer. You may see warnings and that is okay, but you should not see any CMake errors.

A terminal window titled 'build -- zsh -- 89x27' with the path '/Users/bienz/Desktop/Homework-Repo-Template/build -- zsh'. It shows the output of the 'cmake ..' command, including deprecation warnings and configuration details.

```
Compatibility with CMake < 3.10 will be removed from a future version of
CMake.

Update the VERSION argument <min> value. Or, use the <min>...<max> syntax
to tell CMake that the project requires at least <min> but has been updated
to work with policies introduced by <max> or earlier.

CMake Deprecation Warning at build/_deps/googletest-src/googletest/CMakeLists.txt:49 (cma
ke_minimum_required):
Compatibility with CMake < 3.10 will be removed from a future version of
CMake.

Update the VERSION argument <min> value. Or, use the <min>...<max> syntax
to tell CMake that the project requires at least <min> but has been updated
to work with policies introduced by <max> or earlier.

-- Found Python: /opt/homebrew/Frameworks/Python.framework/Versions/3.12/bin/python3.12 (
found version "3.12.7") found components: Interpreter
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Configuring done (3.1s)
-- Generating done (0.0s)
-- Build files have been written to: /Users/bienz/Desktop/Homework-Repo-Template/build
bienz@amandas-mbp build %
```

- (2.4) When you compile your code, you should see files being built and linked. You should also initially see an error at the end. After you create your new function (step 2.5), and add it to the CMakeLists.txt (step 2.7), this error should go away.

```

bienz@amandas-mbp build % make
[  8%] Building CXX object CMakeFiles/homework.dir/Homework-Source-Template/src.cpp.o
[ 16%] Linking CXX static library libhomework.a
[ 16%] Built target homework
[ 25%] Building CXX object _deps/googletest-build/googletest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 33%] Linking CXX static library ../../lib/libgtest.a
[ 33%] Built target gtest
[ 41%] Building CXX object _deps/googletest-build/googlemock/CMakeFiles/gmock.dir/src/gmock-all.cc.o
[ 50%] Linking CXX static library ../../lib/libgmock.a
[ 50%] Built target gmock
[ 58%] Building CXX object _deps/googletest-build/googlemock/CMakeFiles/gmock_main.dir/src/gmock_main.cc.o
[ 66%] Linking CXX static library ../../lib/libgmock_main.a
[ 66%] Built target gmock_main
[ 75%] Building CXX object _deps/googletest-build/googletest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.o
[ 83%] Linking CXX static library ../../lib/libgtest_main.a
[ 83%] Built target gtest_main
[ 91%] Building CXX object Homework-Source-Template/tests/CMakeFiles/unit_tests.dir/unit_tests.cpp.o
[100%] Linking CXX executable unit_tests
Undefined symbols for architecture arm64:
  "return0()", referenced from:
    TLBTest_TestsIntests_Test::TestBody() in unit_tests.cpp.o
    TLBTest_TestsIntests_Test::TestBody() in unit_tests.cpp.o
ld: symbol(s) not found for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
make[2]: *** [Homework-Source-Template/tests/unit_tests] Error 1
make[1]: *** [Homework-Source-Template/tests/CMakeFiles/unit_tests.dir/all] Error 2
make: *** [all] Error 2
bienz@amandas-mbp build %

```

(2.5) There are many options for creating a new file, including:

- ‘touch newfilename.cpp’ will create a file that you can then edit with the editor of your choice.
- ‘vi newfilename.cpp’ will create a new file and open it with VI, if installed.
- ‘emacs newfilename.cpp’ will create a new file and open it with Emacs, if installed.
- Mac users: ‘mvim newfilename.cpp’ will create a new file and open it with MacVim, if installed, allowing for both standard text editing and VI commands.

(2.7) This is what the CMakeLists.txt looks like. If this is not what you see, you have opened the incorrect file. Make sure you are opening the file CMakeLists.txt that is located within the main directory (not a file by the same name within the submodule). **Add the .cpp file that you created either directly above or directly below the SRC_SOURCES variable.**

```

# Will Include Files in Current Directory
set(hw_DIR ${CMAKE_CURRENT_SOURCE_DIR})
set(src_DIR ${CMAKE_CURRENT_SOURCE_DIR}/${SOURCE_NAME})

#####
## GOOGLETEST ##
#####
include(FetchContent)
if (CMAKE_VERSION VERSION_GREATER_EQUAL 3.5)
  set(GTEST_TAG 58d77fa8070e8cec2dc1ed015d66b454c8d78850)
else()
  set(GTEST_TAG e2239ee6043f73722e7aa812a459f54a28552929)
endif()
include(FetchContent)
FetchContent_Declare(
  googletest
  GIT_REPOSITORY https://github.com/google/googletest.git
  # Specify the commit you depend on and update it regularly.
  GIT_TAG ${GTEST_TAG}
)
# For Windows: Prevent overriding the parent project's compiler/linker settings
set(gtest_force_shared_crt ON CACHE BOOL "" FORCE)
FetchContent_MakeAvailable(googletest)
enable_testing()
#####

include_directories(${src_DIR})
include_directories(${hw_DIR})

# Add SRC Directory with Professor's Starter Code
add_subdirectory(${SOURCE_NAME})

# Create Library
# TODO : Include your new file here!
add_library(homework STATIC
  ${src_SOURCES}
)

# Add Directory with Professor's Unit Tests
add_subdirectory(${SOURCE_NAME}/tests)

```

(3.2) Below is what the output of typing ‘make test’ should look like. If this is not what you see, check out common issues in the subsection below.

```

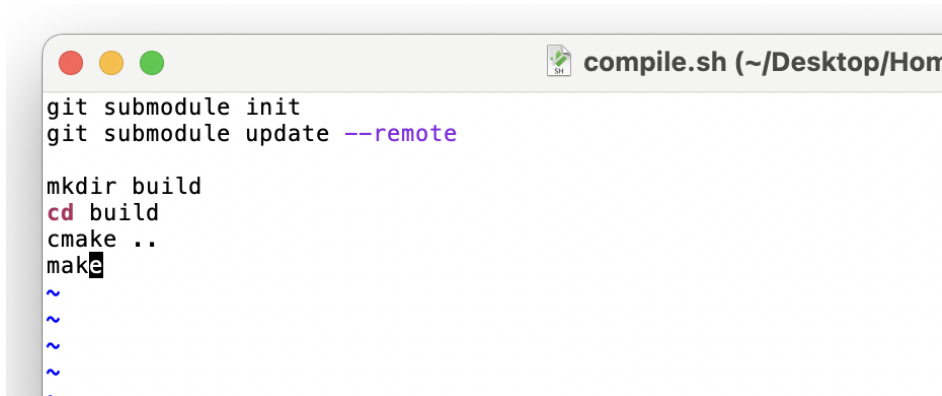
/Users/bienz/Desktop/Homework-Repo-Template/build --zsh
bienz@amandas-mbp build % make test
Running tests...
Test project /Users/bienz/Desktop/Homework-Repo-Template/build
Start 1: UnitTests
1/1 Test #1: UnitTests ..... Passed    0.29 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =  0.30 sec
bienz@amandas-mbp build %

```

(3.4) This is what your ‘compile.sh’ script should look like.



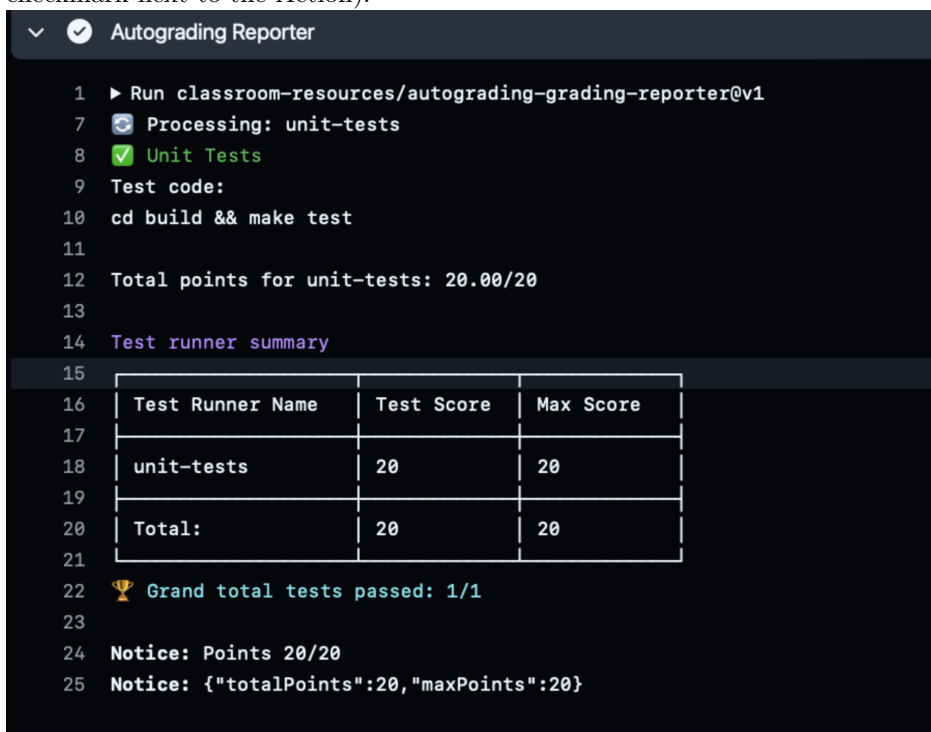
```

git submodule init
git submodule update --remote

mkdir build
cd build
cmake ..
make

```

(4.8) This is what you should see if you open up a passing GitHub actions test case (along with a green checkmark next to the Action).



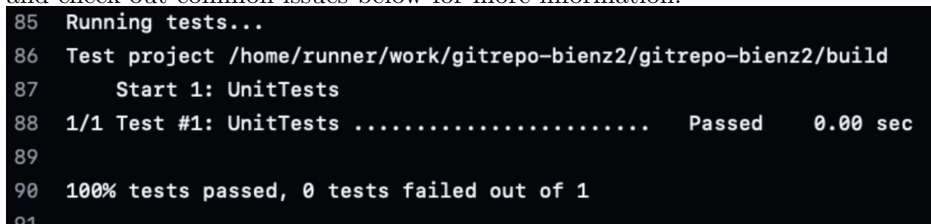
```

✓ Autograding Reporter

1 ▶ Run classroom-resources/autograding-grading-reporter@v1
7 🔄 Processing: unit-tests
8 ✓ Unit Tests
9 Test code:
10 cd build && make test
11
12 Total points for unit-tests: 20.00/20
13
14 Test runner summary
15
16 | Test Runner Name | Test Score | Max Score |
17 |-----|-----|-----|
18 | unit-tests      | 20         | 20         |
19 |-----|-----|-----|
20 | Total:          | 20         | 20         |
21 |-----|-----|-----|
22 🏆 Grand total tests passed: 1/1
23
24 Notice: Points 20/20
25 Notice: {"totalPoints":20,"maxPoints":20}

```

If you click on the the dropdown arrow above the Autograding reporter, you should see the following. If your test is not passing, you should click on the drop-down arrow to see why you are not passing, and check out common issues below for more information.



```

85 Running tests...
86 Test project /home/runner/work/gitrepo-bienz2/gitrepo-bienz2/build
87 Start 1: UnitTests
88 1/1 Test #1: UnitTests ..... Passed    0.00 sec
89
90 100% tests passed, 0 tests failed out of 1
91

```

Common Issues

1. If you get an error while configuring or compiling the library:

- This may be due to an old cache. Try removing everything from within the build directory, and

Cont.

then rerun your configure and compile commands.

- An error during the command 'cmake ..' means there is an issue with your configuration. Make sure you are running this command from the build directory and that the CMakeLists.txt file is in the parent directory. If you still see an error, double check step 2.7.
- An error during the command 'make' means there is an issue compiling your code. Make sure you have a C/C++ compiler installed on your computer. If you still see an error, double check step 2.6.

2. If you get an error while running 'make test':

- If you see tests similar to the image in the Hints above, check that your method 'return0' returns the number 0.
- If you do not see 'UnitTests' running, make sure you are running 'make test' from within your build folder.
- If you still see an error, try running 'make' and double check that your code compiles without error.
- If all else fails, try deleting and recreating your build directory.

3. If you get an error in your GitHub Actions, click on the dropdown arrow for the individual tests (above your autograding report). Some things you may find:

- (a) 'CMake Error' will point to the line within the 'CMakeLists.txt' that an error is encountered. If this includes 'add_subdirectory', the issue is likely your submodule. Go back and check that the code from Part 1, step 5 was added to your 'compile.sh' file.

```
60 CMake Error at CMakeLists.txt:48 (add_subdirectory):
61   The source directory
62
63     /home/runner/work/git-cmake-tutorial-bien2/git-cmake
64
65   does not contain a CMakeLists.txt file.
66
67
```

- (b) 'Cannot find source file' means you are trying to compile code but cannot find the file. Make sure you have added/committed/pushed the new file that you created.

```
64 -- Configuring done (4.7s)
65 Cannot find source file:
66
67   code.cpp
68
69 Tried extensions .c .C .cpp .cc .cxx .cu .m .mp .ms .m
70   .ccm .cxxm .c++m .h .hh .h++ .hm .hpp .hxx .in .txx .
71   .f95 .f03 .hip .ispc
72
73
74 CMake Error at CMakeLists.txt:52 (add_library):
75   No SOURCES given to target: homework
76
```

- (c) If your unit test fails, your return0 function does not return the number 0.

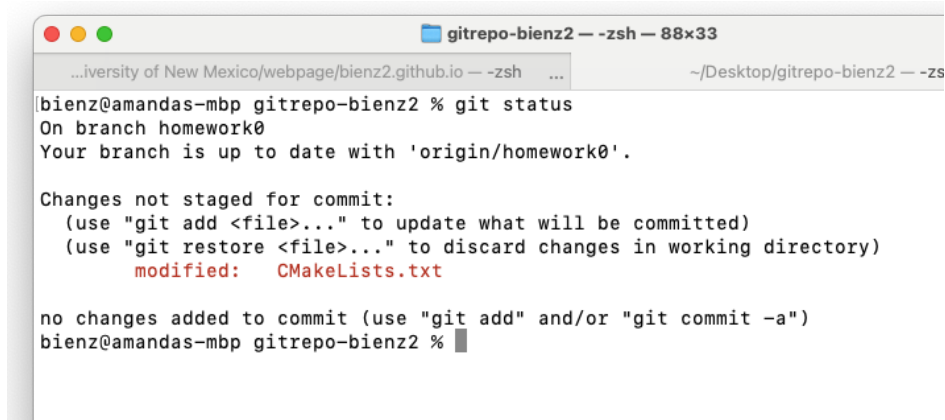
```

85 Running tests...
86 Test project /home/runner/work/gitrepo-bienz2/gitrepo-bienz2/build
87 Start 1: UnitTests
88 1/1 Test #1: UnitTests .....***Failed    0.00 sec
89
90 0% tests passed, 1 tests failed out of 1
91 Errors while running CTest
92

```

4. **Files Not Pushed.** If you have a file on your computer and it is not showing up in your online repository, there are many possible reasons. Type 'git status' and check if you see any of the following:

- (a) The image below indicates you have changed the file 'CMakeLists.txt'. Add this updated file, commit, and push.



```

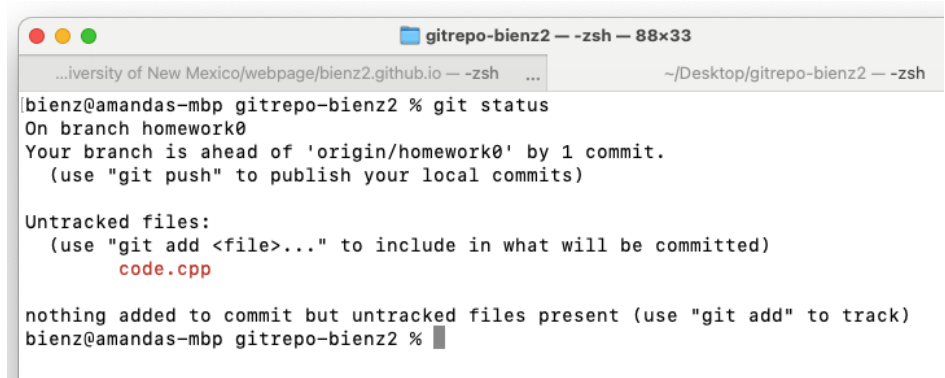
gitrepo-bienz2 -- zsh -- 88x33
...iversity of New Mexico/webpage/bienz2.github.io -- zsh ... ~/Desktop/gitrepo-bienz2 -- zsh
bienz@amandas-mbp gitrepo-bienz2 % git status
On branch homework0
Your branch is up to date with 'origin/homework0'.

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

no changes added to commit (use "git add" and/or "git commit -a")
bienz@amandas-mbp gitrepo-bienz2 %

```

- (b) The image below indicates you have created a new file 'code.cpp'. The command 'git add -u' will not add this file, as it is untracked. You need to type 'git add code.cpp'. Then, you will be able to commit and push the file.



```

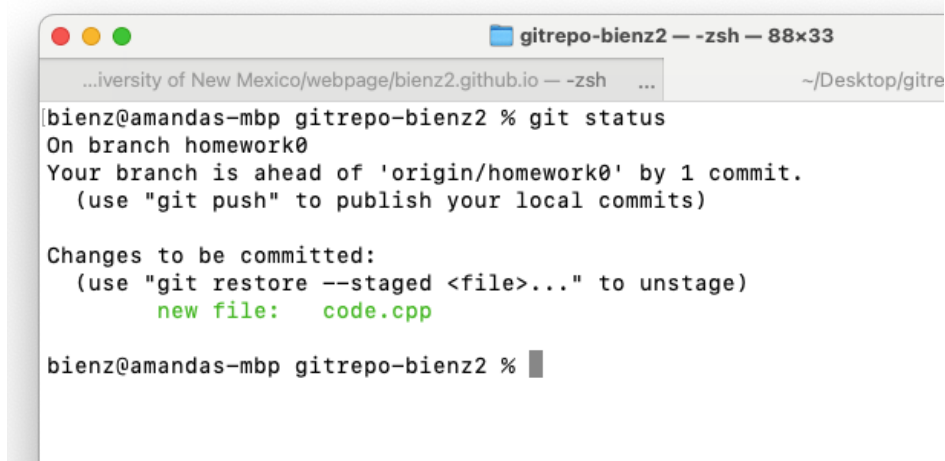
gitrepo-bienz2 -- zsh -- 88x33
...iversity of New Mexico/webpage/bienz2.github.io -- zsh ... ~/Desktop/gitrepo-bienz2 -- zsh
bienz@amandas-mbp gitrepo-bienz2 % git status
On branch homework0
Your branch is ahead of 'origin/homework0' by 1 commit.
  (use "git push" to publish your local commits)

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

nothing added to commit but untracked files present (use "git add" to track)
bienz@amandas-mbp gitrepo-bienz2 %

```

- (c) The image below indicates you have already added files, but have not committed or pushed. Commit and push these files.



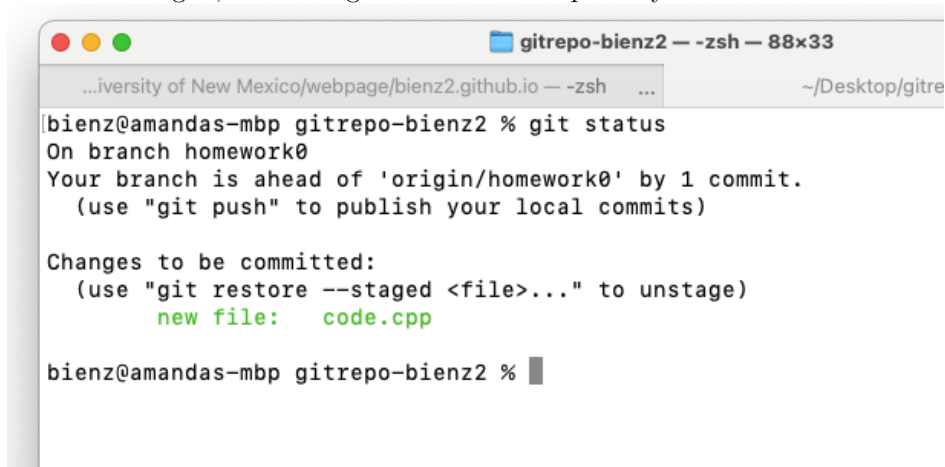
```
gitrepo-bienz2 — -zsh — 88x33
...iversity of New Mexico/webpage/bienz2.github.io — -zsh ... ~/Desktop/gitre

bienz@amandas-mbp gitrepo-bienz2 % git status
On branch homework0
Your branch is ahead of 'origin/homework0' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   code.cpp

bienz@amandas-mbp gitrepo-bienz2 %
```

- (d) The image below indicates you have already committed files, but have not pushed. Push these files. Hint: there are no colors here. The indication that you have committed is that 'your branch is ahead of origin', where 'origin' is the online repository.



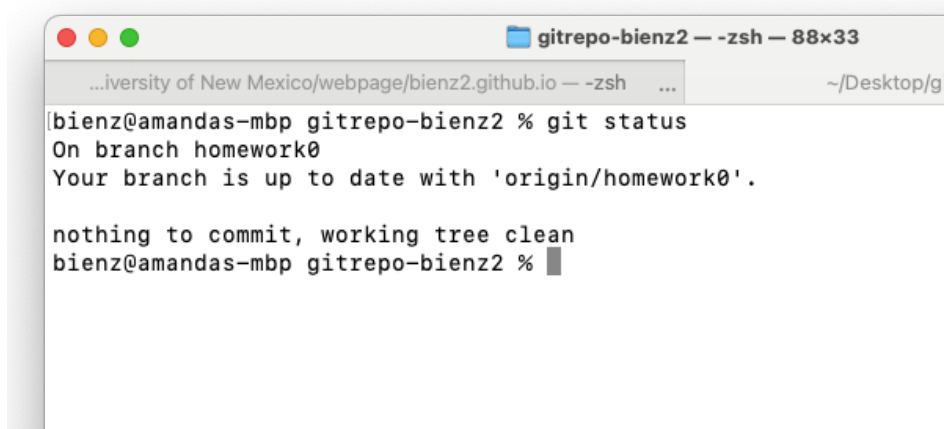
```
gitrepo-bienz2 — -zsh — 88x33
...iversity of New Mexico/webpage/bienz2.github.io — -zsh ... ~/Desktop/gitre

bienz@amandas-mbp gitrepo-bienz2 % git status
On branch homework0
Your branch is ahead of 'origin/homework0' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   code.cpp

bienz@amandas-mbp gitrepo-bienz2 %
```

- (e) If all updated files have been committed and pushed, your repository should look like the following.



```
gitrepo-bienz2 — -zsh — 88x33
...iversity of New Mexico/webpage/bienz2.github.io — -zsh ... ~/Desktop/g

bienz@amandas-mbp gitrepo-bienz2 % git status
On branch homework0
Your branch is up to date with 'origin/homework0'.

nothing to commit, working tree clean
bienz@amandas-mbp gitrepo-bienz2 %
```

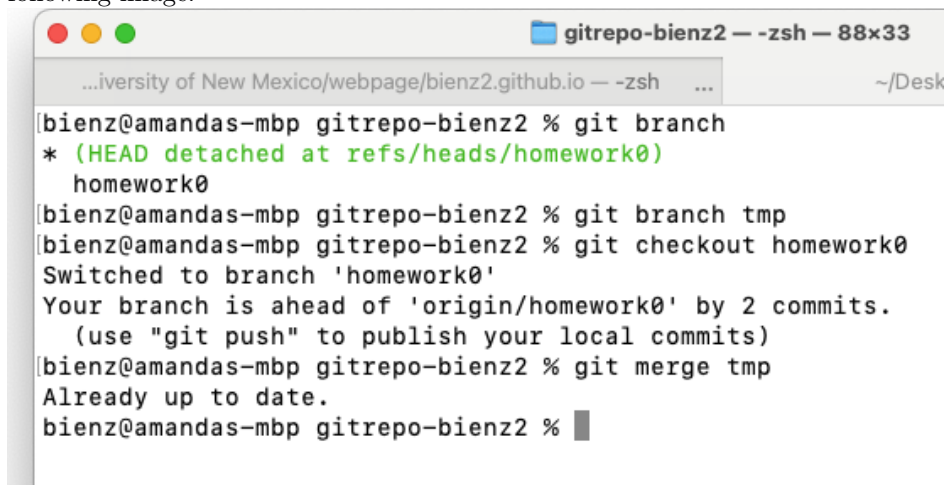
5. **Branch Issues:** If your status shows all files have been pushed, type 'git branch'. This should only show one branch.

- (a) If instead you see HEAD detached, this means you are no longer appropriately connected to your GitHub repository.

A terminal window titled "gitrepo-bienz2" with a subtitle "...iversity of New Mexico/webpage/bienz2.github.io — -zsh". The prompt is "bienz@amandas-mbp gitrepo-bienz2 %". The user enters "git branch", and the output is "* (HEAD detached at refs/heads/homework0)" followed by "homework0" on the next line. The prompt returns to "bienz@amandas-mbp gitrepo-bienz2 %".

```
[bienz@amandas-mbp gitrepo-bienz2 % git branch
* (HEAD detached at refs/heads/homework0)
homework0
bienz@amandas-mbp gitrepo-bienz2 %
```

- (b) To fix this, you can create a new branch and merge it into the correct branch, as shown in the following image.

A terminal window titled "gitrepo-bienz2" with a subtitle "...iversity of New Mexico/webpage/bienz2.github.io — -zsh" and a window size of "88x33". The prompt is "bienz@amandas-mbp gitrepo-bienz2 %". The user enters "git branch", and the output is "* (HEAD detached at refs/heads/homework0)" followed by "homework0". The user then enters "git branch tmp", "git checkout homework0", and "git merge tmp". The output shows the branch creation, switching to homework0, and a successful merge. The prompt returns to "bienz@amandas-mbp gitrepo-bienz2 %".

```
[bienz@amandas-mbp gitrepo-bienz2 % git branch
* (HEAD detached at refs/heads/homework0)
homework0
[bienz@amandas-mbp gitrepo-bienz2 % git branch tmp
[bienz@amandas-mbp gitrepo-bienz2 % git checkout homework0
Switched to branch 'homework0'
Your branch is ahead of 'origin/homework0' by 2 commits.
(use "git push" to publish your local commits)
[bienz@amandas-mbp gitrepo-bienz2 % git merge tmp
Already up to date.
bienz@amandas-mbp gitrepo-bienz2 %
```