(Mac Computers Only) Installing and Setting-Up VS Code

- Updated on 8/28/2023

---

1. Download and install VSCode for Mac via their website: Download Page

2. Open VSCode, and install the **"Remote-SSH" extension** and the **"C++ Extension Pack" extension.**

3. In VSCode, open the terminal and paste the following command; this will install CLang, something required in order to run C++ programs.

```
xcode-select --install
```

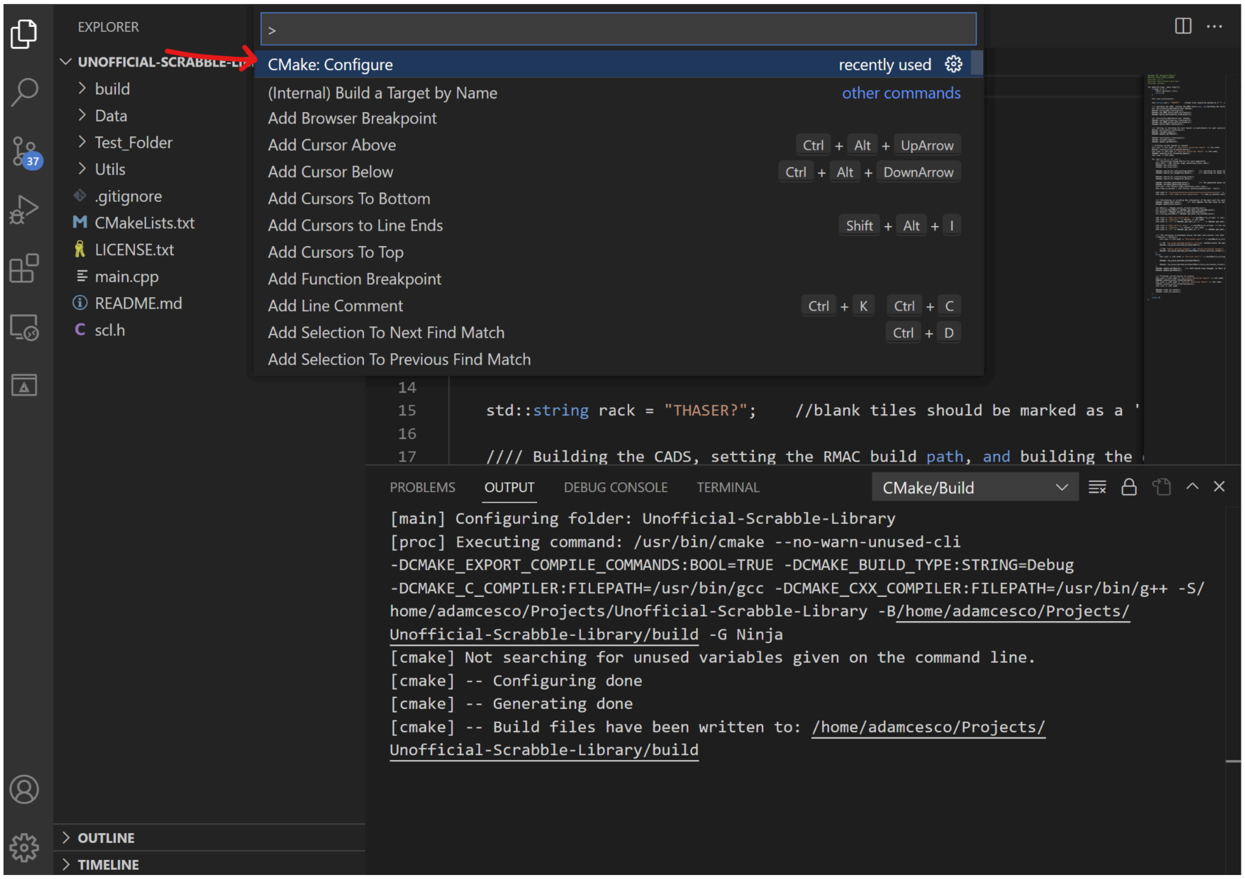
4. Now you should create a new folder named "projects" somewhere on your Mac computer, this is where you should store all your C++ projects for the rest of the class.

5. Then we are going to create a new folder named "HelloWorld" within the "projects" folder you just created in the previous step. This "HelloWorld" folder is going to host a new project where we will run code to make sure your environment is set up correctly.

6. Open VSCode again, then open the "HelloWorld" folder in VSCode and create a main.cpp file. Write some basic hello-world code in that new main.cpp file.

7. Download this **file** and add it to this project, so that your project has a CMakeLists.txt file in addition to a main.cpp file. You can add this file to your project by downloading the file and then dragging and dropping it next to the main.cpp file in VSCode.

8. Press "COMMAND + SHIFT + P", and then select the "CMake: Configure" option, shown below:

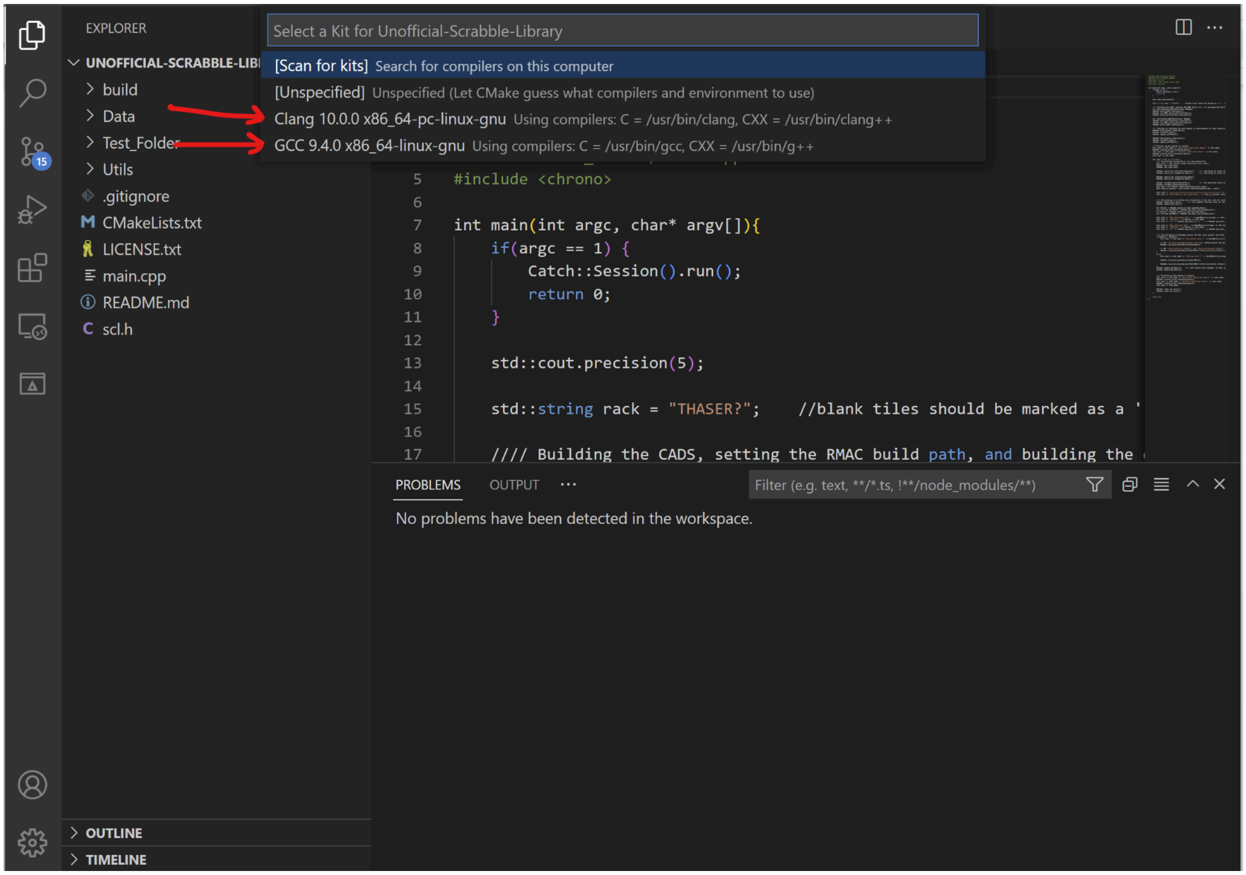


```

    std::string rack = "THASER?";    //blank tiles should be marked as a '

    //// Building the CADS, setting the RMAC build path, and building the
```

```
[main] Configuring folder: Unofficial-Scrabble-Library
[proc] Executing command: /usr/bin/cmake --no-warn-unused-cli
-DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE -DCMAKE_BUILD_TYPE:STRING=Debug
-DCMAKE_C_COMPILER:FILEPATH=/usr/bin/gcc -DCMAKE_CXX_COMPILER:FILEPATH=/usr/bin/g++ -S/
home/adamcesco/Projects/Unofficial-Scrabble-Library -B/home/adamcesco/Projects/
Unofficial-Scrabble-Library/build -G Ninja
[cmake] Not searching for unused variables given on the command line.
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /home/adamcesco/Projects/
Unofficial-Scrabble-Library/build
```

- After doing this, you are going to get a lot of pop-ups and notifications, and most likely they are not going to be in the same order as below. If you ever get lost and need to start over, just press "COMMAND + SHIFT + P" again, and repeat Step 8 and on.

9. When you get a "Select a Kit" prompt shown below, select either the option that says "Clang" or "GCC"


EXPLORER
UNOFFICIAL-SCRABBLE-LIB
build
Data
Test_Folder
Utils
.gitignore
CMakeLists.txt
LICENSE.txt
main.cpp
README.md
scl.h
OUTLINE
TIMELINE
15
Select a Kit for Unofficial-Scrabble-Library
[Scan for kits] Search for compilers on this computer
[Unspecified] Unspecified (Let CMake guess what compilers and environment to use)
Clang 10.0.0 x86_64-pc-linux-gnu Using compilers: C = /usr/bin/clang, CXX = /usr/bin/clang++
GCC 9.4.0 x86_64-linux-gnu Using compilers: C = /usr/bin/gcc, CXX = /usr/bin/g++
PROBLEMS
OUTPUT
Filter (e.g. text, **/*.ts, !**/node_modules/**)

```
#include <chrono>

int main(int argc, char* argv[]){
    if(argc == 1) {
        Catch::Session().run();
        return 0;
    }

    std::cout.precision(5);

    std::string rack = "THASER?";    //blank tiles should be marked as a '

    //// Building the CADS, setting the RMAC build path, and building the
```

No problems have been detected in the workspace.

- If "GCC" does not work or is not an option, select "Clang" (and vice versa)
- If you do not get this "Select a Kit" prompt, click the button that says "No Kit Selected" or "No active kit" in the blue bottom toolbar, and choose "Clang" or "GCC"

10. **After selecting a kit the configuration process will continue. It is very common to get an error** that appears in the bottom right corner of the VSCode window. **The error should state something like "Bad CMake executable path".** If you did not get this error, skip to Step 22. If you did get this error, continue to the next step. If you get stuck on fixing this issue, please reach out to one of the TAs via Slack or email!

11. You have the "Bad CMake executable path" error because you either do not have CMake installed or some settings within the "CMake Tools" VSCode extension are incorrect, both are very fixable! **Install homebrew by copying and pasting the following command into the VSCode terminal:**

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

12. After this command has fully completed, the terminal should tell you two different commands to copy and paste into your terminal. The two commands should be on different lines and should look something like the following commands. **DO NOT ENTER THESE EXACT COMMANDS, COPY THE ACTUAL COMMANDS STATED IN YOUR TERMINAL**:

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"'
eval "$(/opt/homebrew/bin/brew shellenv)"
```

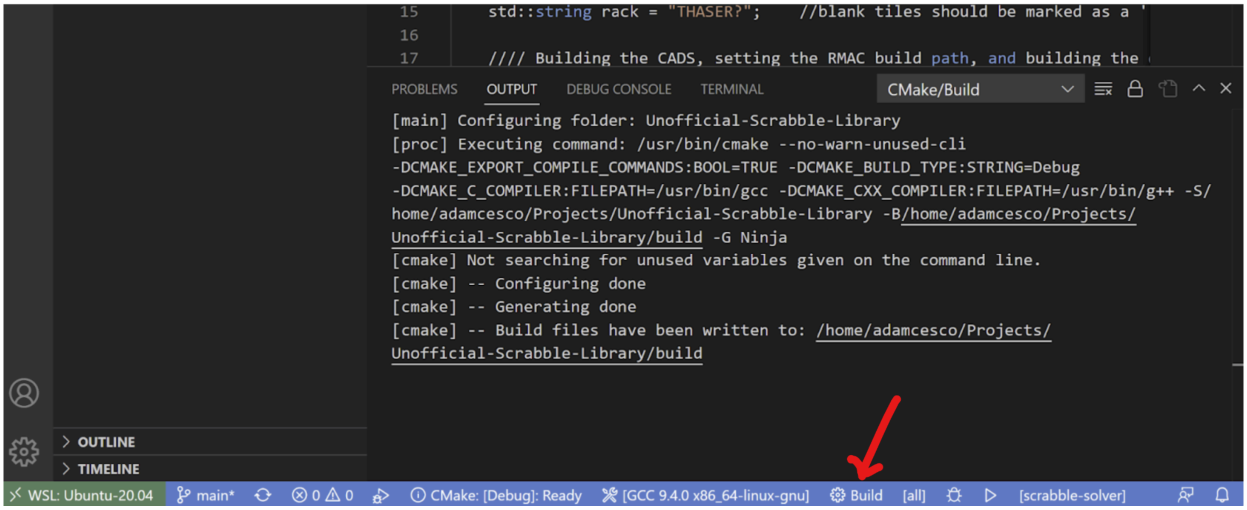
13. Now homebrew is installed correctly. **Now enter the following command into your VSCode terminal**, if this command does not work and produces an error, then go back to Step 11 or contact a TA via Slack or email.

```
brew install cmake
```

14. Now we need to check if CMake is installed correctly. **Enter the following command into your VSCode terminal**, the command should list the versions of CMake that you have installed on your Mac. If CMake is not found or a version is not stated, then CMake was not installed correctly and you should contact a TA via Slack or email.

```
cmake --version
```

15. Now CMake should be properly installed. Try to click the "build" button on the blue bottom toolbar shown below to see if you still get that "Bad CMake executable path" error. If you did not get this error, skip to Step 20. If you did get this error, continue to the next step (Step 16).



```
    std::string rack = "THASER?";    //blank tiles should be marked as a

    //// Building the CADS, setting the RMAC build path, and building the
```

```
[main] Configuring folder: Unofficial-Scrabble-Library
[proc] Executing command: /usr/bin/cmake --no-warn-unused-cli
-DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE -DCMAKE_BUILD_TYPE:STRING=Debug
-DCMAKE_C_COMPILER:FILEPATH=/usr/bin/gcc -DCMAKE_CXX_COMPILER:FILEPATH=/usr/bin/g++ -S/
home/adamcesco/Projects/Unofficial-Scrabble-Library -B/home/adamcesco/Projects/
Unofficial-Scrabble-Library/build -G Ninja
[cmake] Not searching for unused variables given on the command line.
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /home/adamcesco/Projects/
Unofficial-Scrabble-Library/build
```

16. At this point, it is common that you are still getting the "Bad CMake executable path" error because CMake is installed correctly but there are still some settings in the "CMake Tools" extension that are incorrect, very fixable. **Copy and paste the following command into the VSCode terminal:**

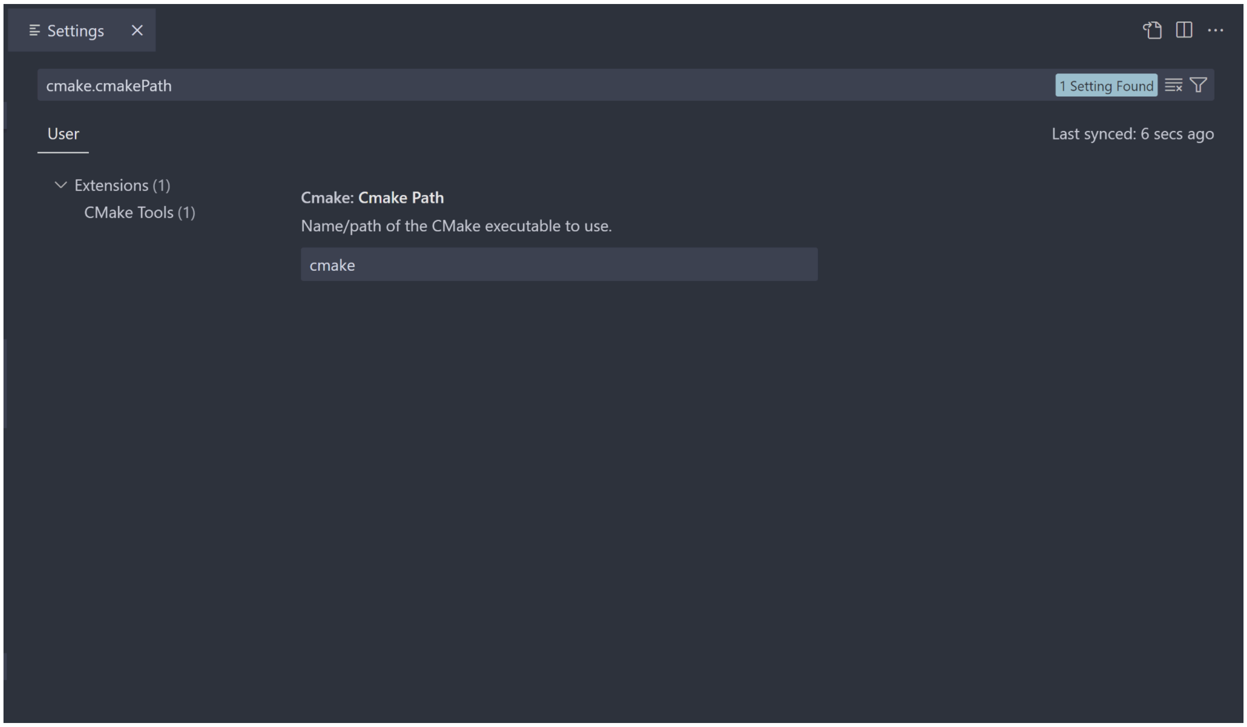
```
which cmake
```

17. **Copy the output of that terminal command**; it should look like "opt/brew/username/cmake" or something similar.

18. Now open your VSCode settings and type the following into the settings search bar:

```
cmake.cmakePath
```

19. Your screen should look like such:



cmake.cmakePath

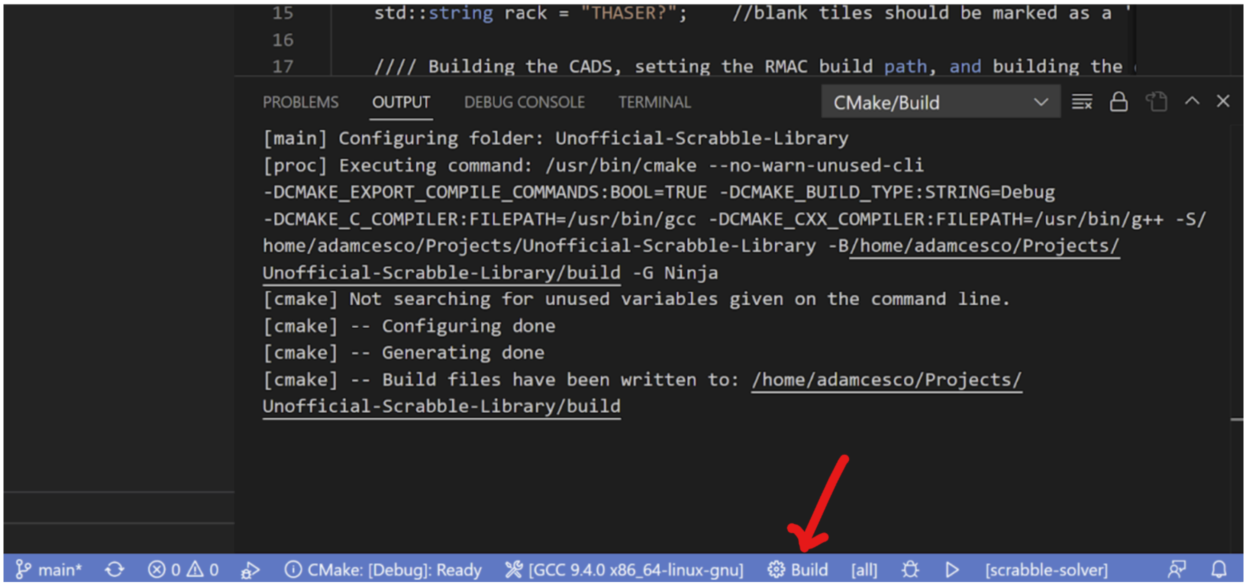
Cmake: Cmake Path

Name/path of the CMake executable to use.

cmake

20. In the text-box for the "Cmake Path" setting, change the text from "cmake" to the terminal output you copied in Step 16 and 17.

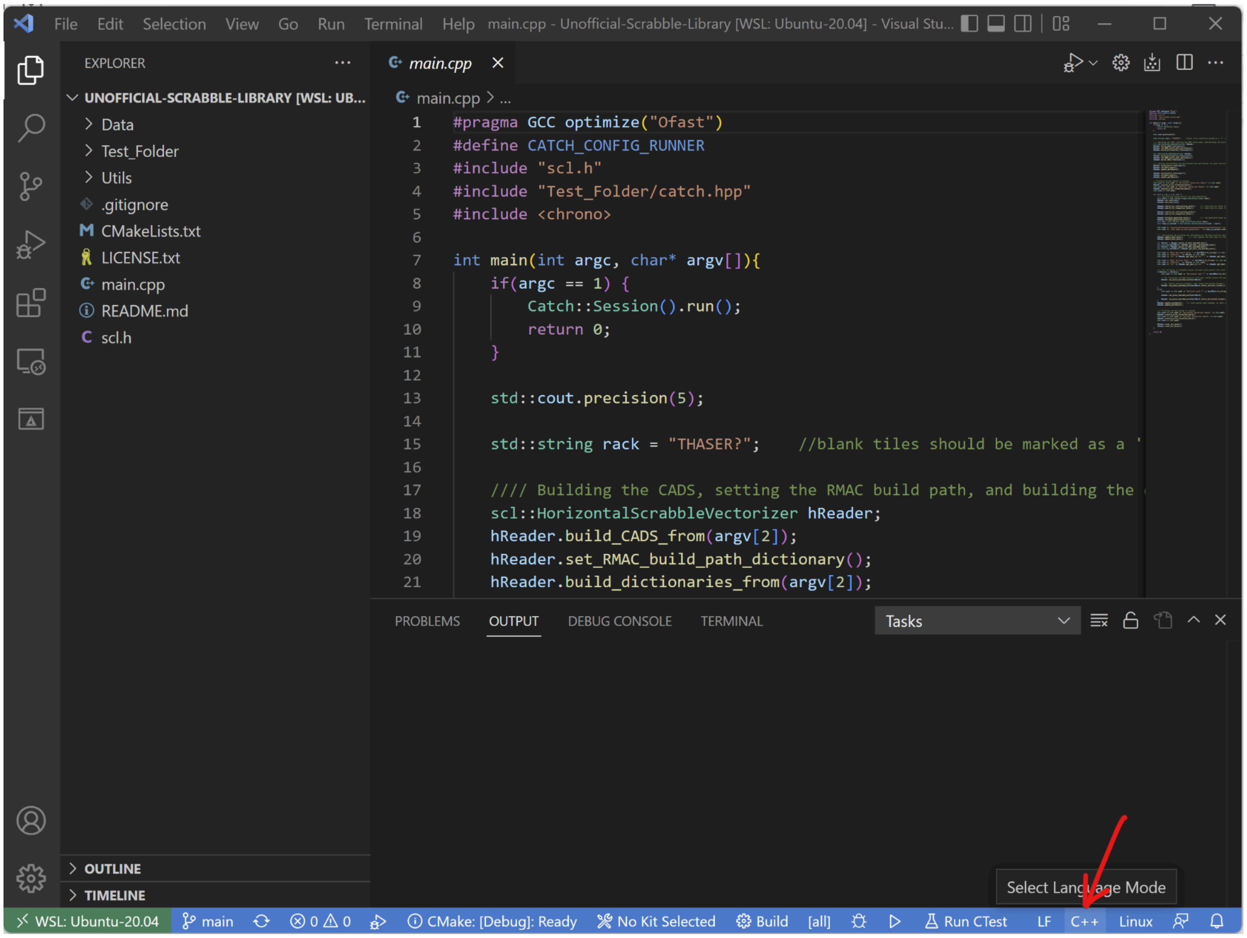
21. Now CMake should be properly installed. Try to click the "build" button on the blue bottom toolbar shown below to see if you still get that "Bad CMake executable path" error. **If you still get this error, contact a TA via Slack or email.** If you did not get this error, continue to the next step.



```
    std::string rack = "THASER?";    //blank tiles should be marked as a '

    //// Building the CADS, setting the RMAC build path, and building the
```

```
[main] Configuring folder: Unofficial-Scrabble-Library
[proc] Executing command: /usr/bin/cmake --no-warn-unused-cli
-DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE -DCMAKE_BUILD_TYPE:STRING=Debug
-DCMAKE_C_COMPILER:FILEPATH=/usr/bin/gcc -DCMAKE_CXX_COMPILER:FILEPATH=/usr/bin/g++ -S/
home/adamcesco/Projects/Unofficial-Scrabble-Library -B/home/adamcesco/Projects/
Unofficial-Scrabble-Library/build -G Ninja
[cmake] Not searching for unused variables given on the command line.
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /home/adamcesco/Projects/
Unofficial-Scrabble-Library/build
```

22. If your current VSCode window does not say C++ where the red arrow is pointing below, click on the button the red arrow is pointing to and select the "C++" option



File Edit Selection View Go Run Terminal Help main.cpp - Unofficial-Scrabble-Library [WSL: Ubuntu-20.04] - Visual Stu...
EXPLORER
UNOFFICIAL-SCRABBLE-LIBRARY [WSL: UB...
Data
Test_Folder
Utils
.gitignore
CMakeLists.txt
LICENSE.txt
main.cpp
README.md
scl.h
OUTLINE
TIMELINE
main.cpp
main.cpp > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Tasks
Select Language Mode
WSL: Ubuntu-20.04 main 0 0 CMake: [Debug]: Ready No Kit Selected Build [all] Run CTest LF C++ Linux

```
#pragma GCC optimize("Ofast")
#define CATCH_CONFIG_RUNNER
#include "scl.h"
#include "Test_Folder/catch.hpp"
#include <chrono>

int main(int argc, char* argv[]){
    if(argc == 1) {
        Catch::Session().run();
        return 0;
    }

    std::cout.precision(5);

    std::string rack = "THASER?";    //blank tiles should be marked as a '

    //// Building the CADS, setting the RMAC build path, and building the
    scl::HorizontalScrabbleVectorizer hReader;
    hReader.build_CADS_from(argv[2]);
    hReader.set_RMAC_build_path_dictionary();
    hReader.build_dictionaries_from(argv[2]);
```

- If your window does not say anything where the arrow is pointing, then don't worry about it and move to the next step

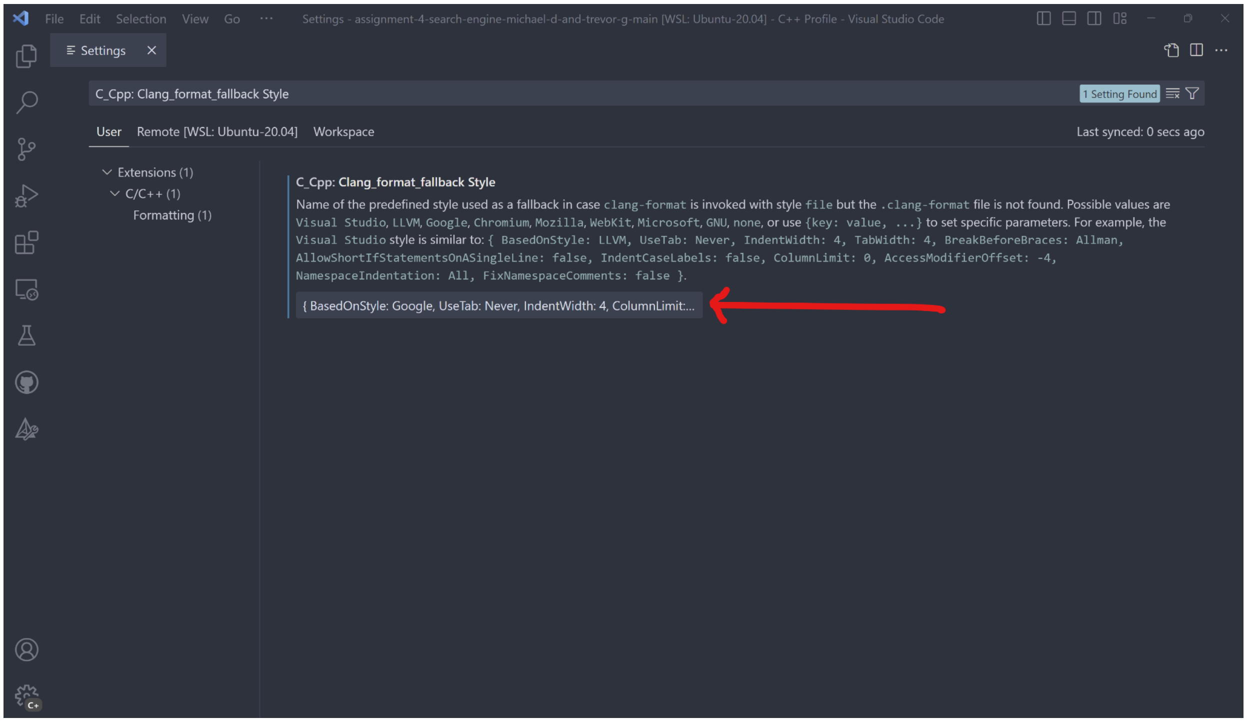
23. Your "HelloWorld" project should now work in VSCode as intended. Run your code with the triangle run button on the blue bottom toolbar. **Never run your code with the triangle run button in the top right.**

24. Now we are going to edit your formatting options in VS Code, so that **your code is nicely formatted when you press the "COMMAND + K + F" command**. Go to your VS Code settings and paste the following highlighted text into the settings search bar:

C_Cpp: Clang_format_fallback Style

25. Now in the text-box pointed to by the red arrow in the picture below, delete the text currently there, and replace it with the highlighted text below:

{ BasedOnStyle: Google, UseTab: Never, IndentWidth: 4, ColumnLimit: 0 }


File
Edit
Selection
View
Go
···
Settings - assignment-4-search-engine-michael-d-and-trevor-g-main [WSL: Ubuntu-20.04] - C++ Profile - Visual Studio Code
Settings
1 Setting Found
User
Remote [WSL: Ubuntu-20.04]
Workspace
Last synced: 0 secs ago
Extensions (1)
C/C++ (1)
Formatting (1)

C_Cpp: Clang_format_fallback Style

C_Cpp: Clang_format_fallback Style

Name of the predefined style used as a fallback in case `clang-format` is invoked with style `file` but the `.clang-format` file is not found. Possible values are `Visual Studio`, `LLVM`, `Google`, `Chromium`, `Mozilla`, `WebKit`, `Microsoft`, `GNU`, `none`, or use `{key: value, ...}` to set specific parameters. For example, the `Visual Studio` style is similar to: `{ BasedOnStyle: LLVM, UseTab: Never, IndentWidth: 4, TabWidth: 4, BreakBeforeBraces: Allman, AllowShortIfStatementsOnASingleLine: false, IndentCaseLabels: false, ColumnLimit: 0, AccessModifierOffset: -4, NamespaceIndentation: All, FixNamespaceComments: false }`.

{ BasedOnStyle: Google, UseTab: Never, IndentWidth: 4, ColumnLimit:...

26. Now you are ready to code for the rest of the semester. Make sure that when you code, you routinely format your code with "COMMAND + K + F" so that you and the TAs can easily read it.