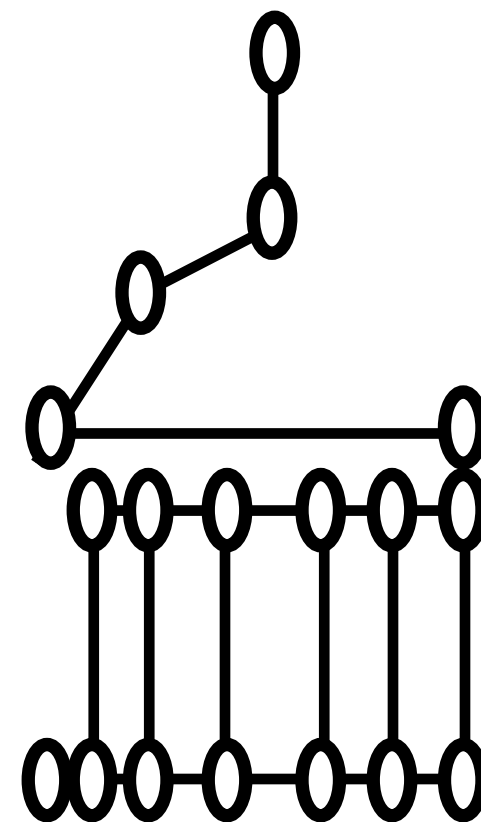


Lecture: CLion, CMake and Debugging

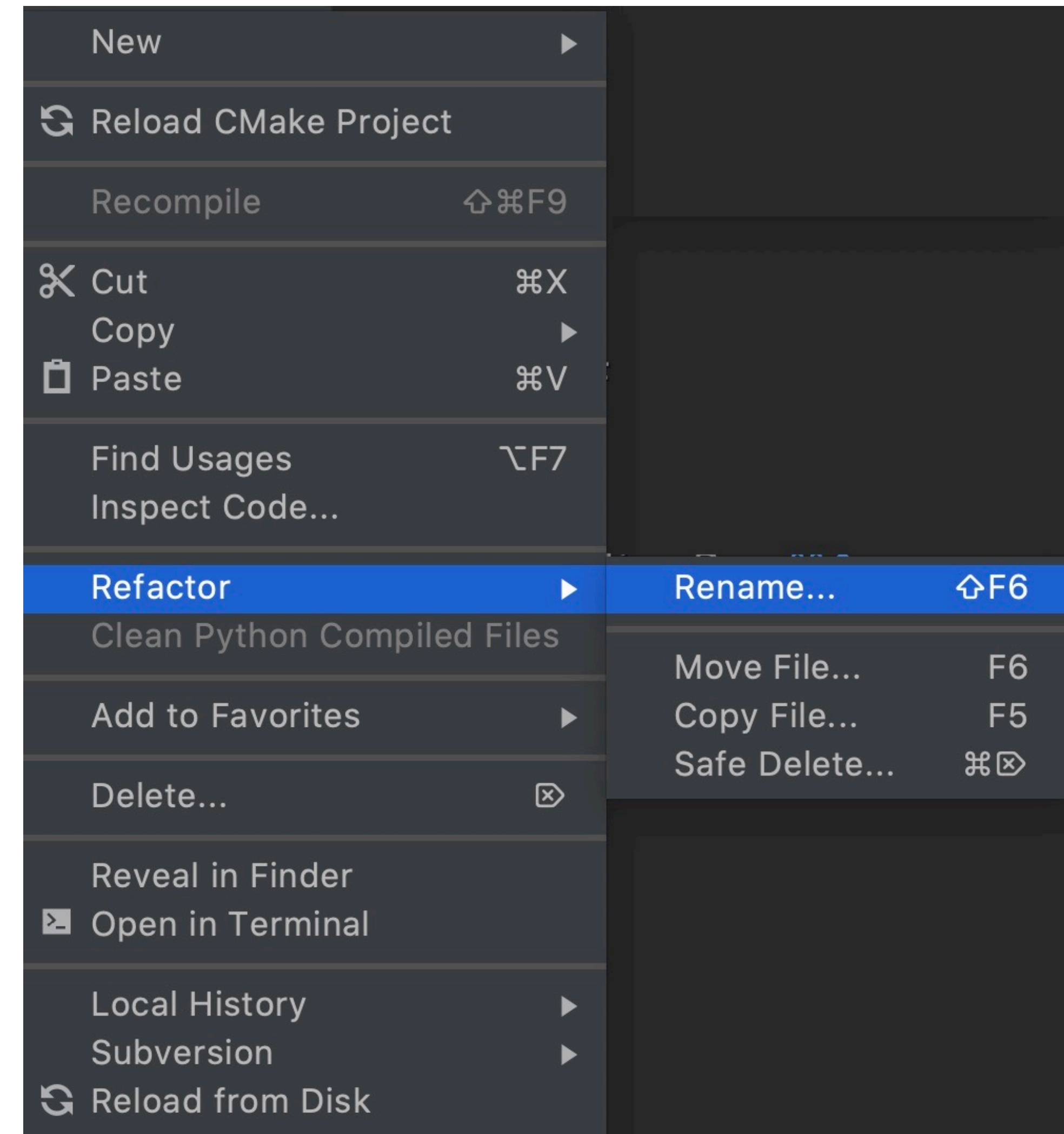
ENGR 2730: Computers in Engineering





Renaming Files in CLion

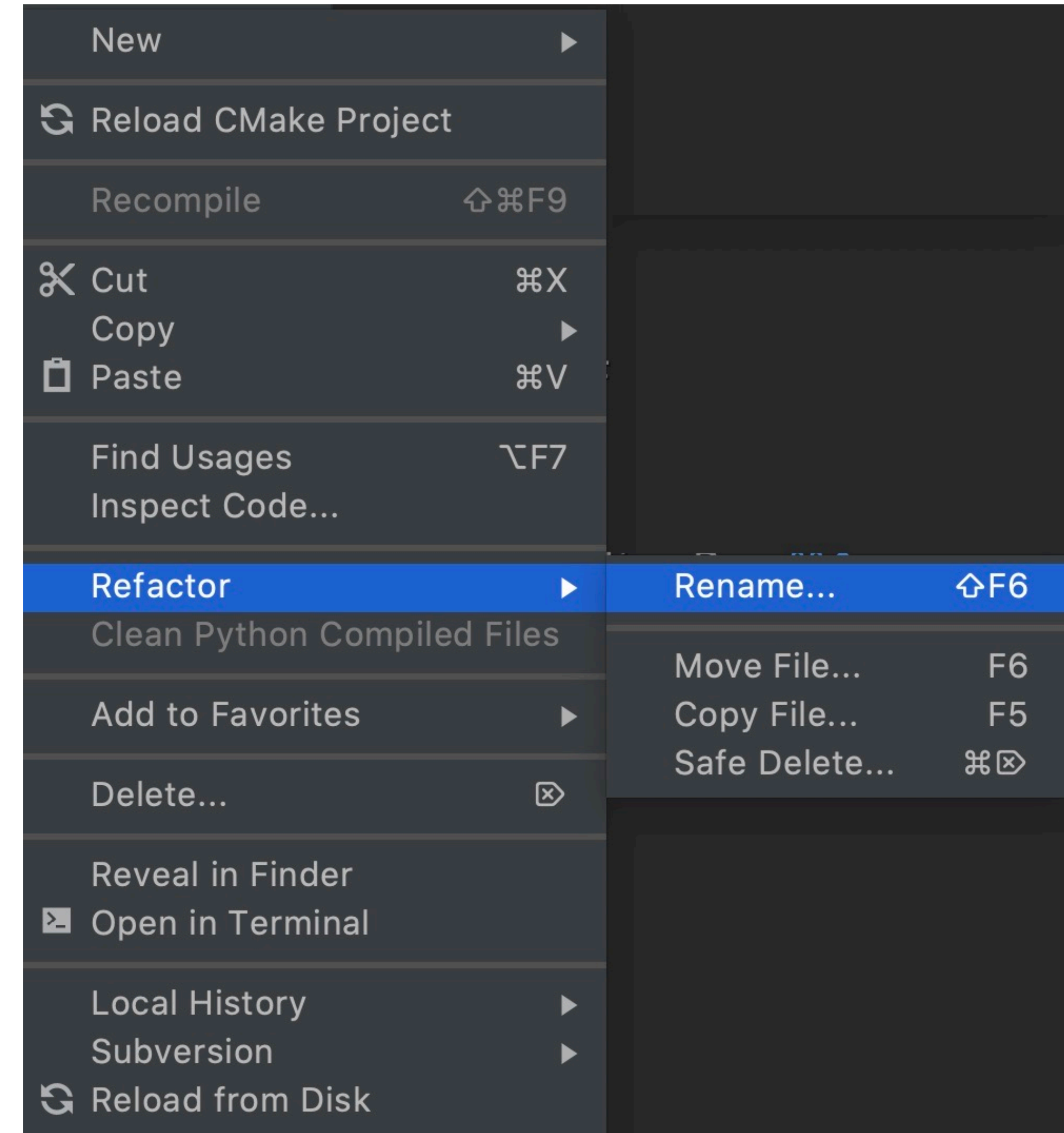
- Use "refactor" to rename a file.
- This changes the name of the file throughout the project
Examples:
 - comments in all the files in the project
 - CMakeLists.txt file.
 - header files
- **DO NOT rename a file with a different case.**
For Example, if you have a file or directory called "HW1" then do not rename it to "hw1".
This will screwup your local repo because the remote repo will have both files and will check out both to the same name.






Renaming Files in CLion (con't)

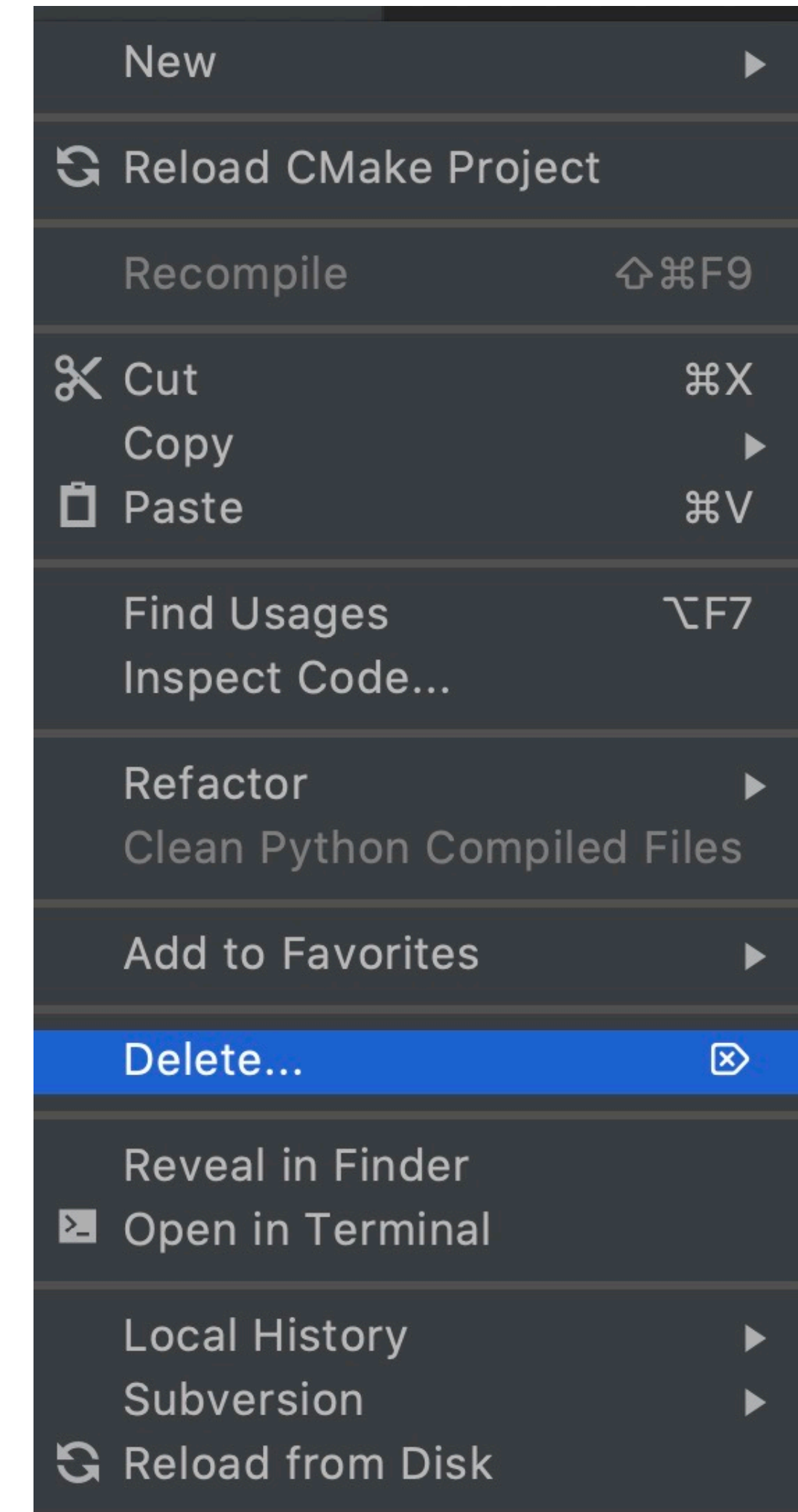
- Renaming a file in your local and remote SVN repos.
- If file is red/tan/grey, then the file is only on your local machine. The "Rename" command will rename the file on your local computer and some file names may turn blue.
- Blue means the file was modified and needs to be committed to the remote repo. Commit the file to the remote repo.
- If file is green, then it was scheduled to be added to the remote repo. Commit the file to the remote repo.
- If file is white/black, then it will turn blue. Then follow steps above.





Deleting Files in CLion and SVN

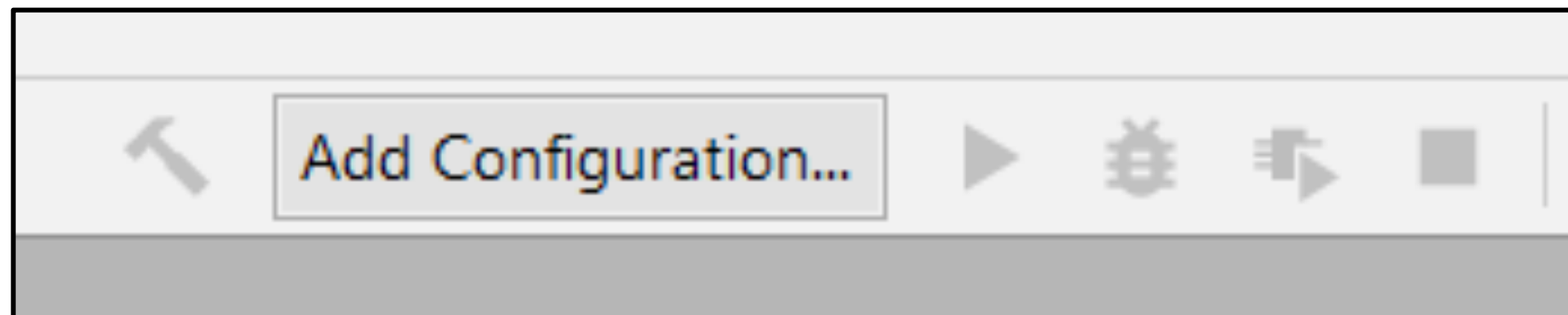
- Use "delete" to remove a file from your local and remote SVN repos.
- If a file is red/tan/grey, then the file is only on your local machine. The "delete" command will delete the file from your local computer. Done.
- If a file is green, then it is scheduled to be added to the remote repo. You must first "revert" the file to unschedule the add. This will change the file to red and you can delete the file as above.
- If a file is blue, then you have made modifications to the file. You must first "revert"  the file to make it white/black. Then delete the file as follows.
- If a file is white/black, then it is in the remote repo. The "delete" command will (1) delete the file from your computer and (2) ask you if you want to delete the file from the remote repo. Agree to delete the file from the remote repo. You must then commit this change to delete the file from remote repo.



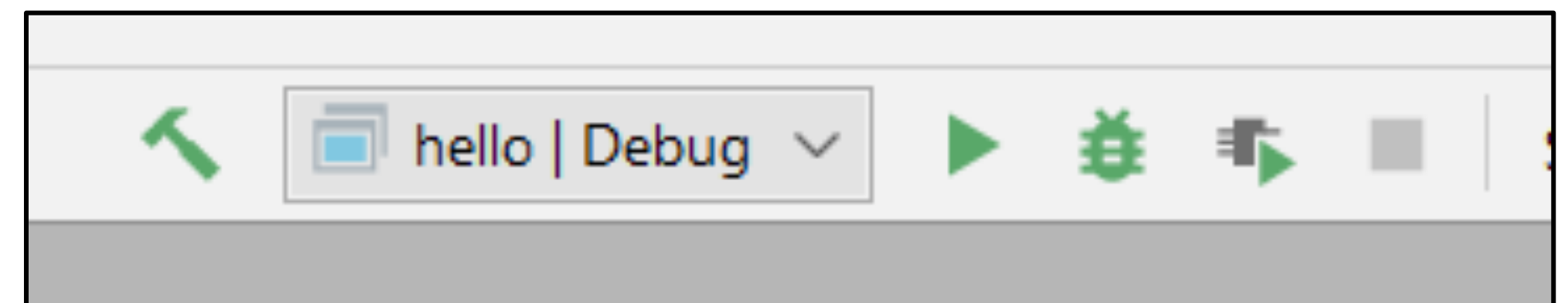
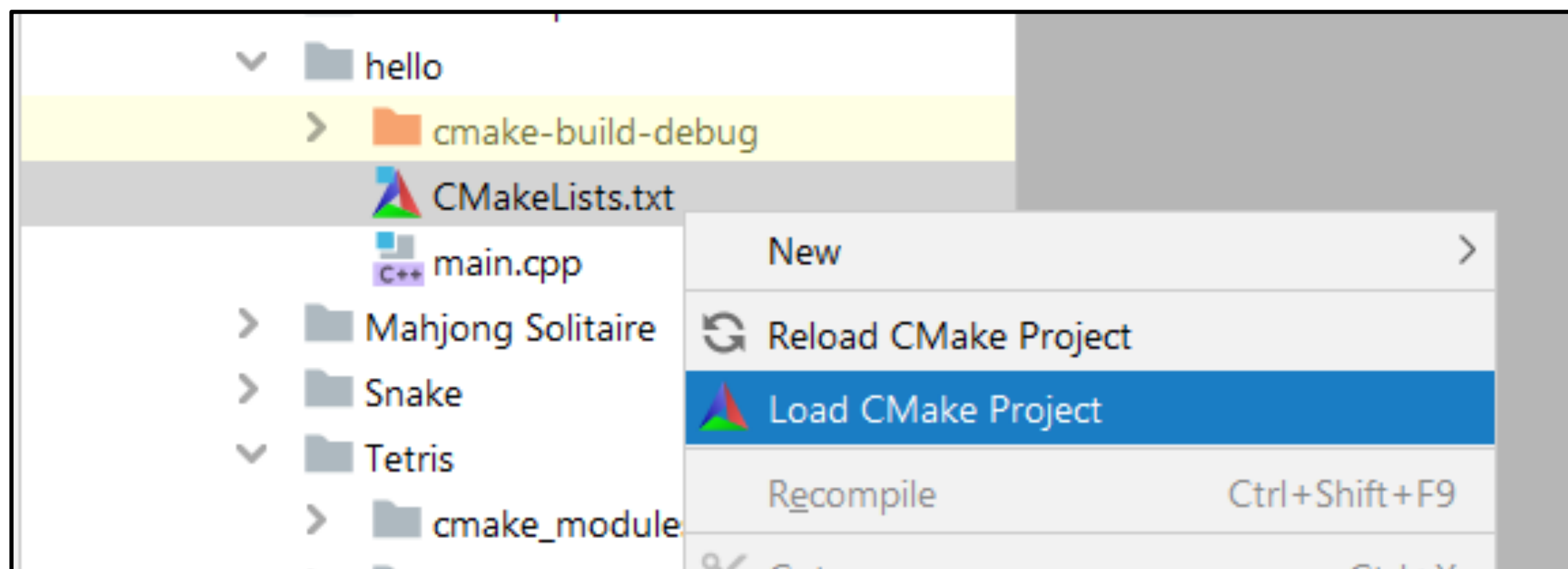


Load a Project in CLion

- If you see “Add Configuration...” there is no active project in CLion



- To load a project: in the project's folder, right click the “CMakeLists.txt” file and select “Load CMake Project”

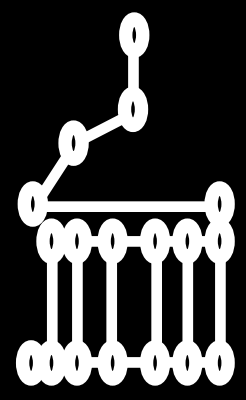


	A Simple Project CMakeLists.txt file
1	cmake_minimum_required(VERSION 3.15)
2	project(hello)
3	
4	set(CMAKE_CXX_STANDARD 14)
5	
6	add_executable(hello main.cpp)

CMake tutorial:

<http://derekmolloy.ie/hello-world-introductions-to-cmake/>

- Line 1 sets the minimum version of CMake for this project
- Line 2 is the *project()* command that sets the project name.
- Line 4 is the minimum version of C++ needed to build the project.
- Line 6 is the *add_executable()* command, which requests that an executable be built using the main.cpp source file. The first argument to the *add_executable()* function is the name of the executable to be built, and the second argument is the source file from which to build the executable.



Try building/running the starting program

Build Run Debug

What will be built (active target)

The screenshot shows an IDE window with the following components:

- Menu Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Toolbar:** Includes icons for Build (green left arrow), Run (green right arrow), and Debug (green bug icon).
- Project Explorer:** Shows a tree view of the project structure. The 'hello' target is selected, and the 'main.cpp' file is highlighted.
- Code Editor:** Displays the contents of 'main.cpp':

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello, Hello" << endl;
8     return 0;
9 }
```

Blue arrows indicate the workflow: 'Build' points to the Build button, 'Run' points to the Run button, and 'Debug' points to the Debug button. A large blue arrow points from the 'hello | Debug' dropdown menu to the text 'What will be built (active target)'.



CLion Debugger Demo

The screenshot shows the CLion IDE interface. At the top, there's a toolbar with a green arrow, a dropdown menu showing 'hello | Debug', a green play button, a green bug icon, and a grey square. A blue arrow points from the text 'Run with Debugging' to the bug icon. Below the toolbar, there are two tabs: 'main.cpp' (active) and 'CMakeLists.txt'. The 'main.cpp' tab shows a C++ file with the following code:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int value = 2112;
8
9      cout << "The value is " << value << endl;
10
11     return 0;
12 }
```

A blue arrow points from the text 'Click in this area to set a debugging "breakpoint"' to a red dot on line 7, which is the line containing 'int value = 2112;'. The line is highlighted in light red. The line number 10 is highlighted in yellow.



CLion Debugger Demo

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7  int value = 2112;
8
9  cout << "The value is " << value << endl;
10
11  return 0;
12 }
```

Debug: hello

Debugger Console

Frames

- main main.cpp:7
- _tmainCRTStartup 0x00000000004013f7
- mainCRTStartup 0x000000000040152b

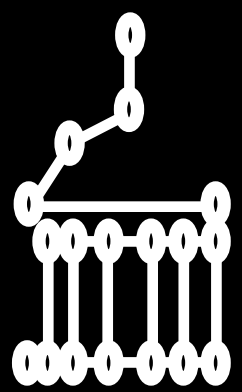
Variables GDB

Step Over
(execute) Line

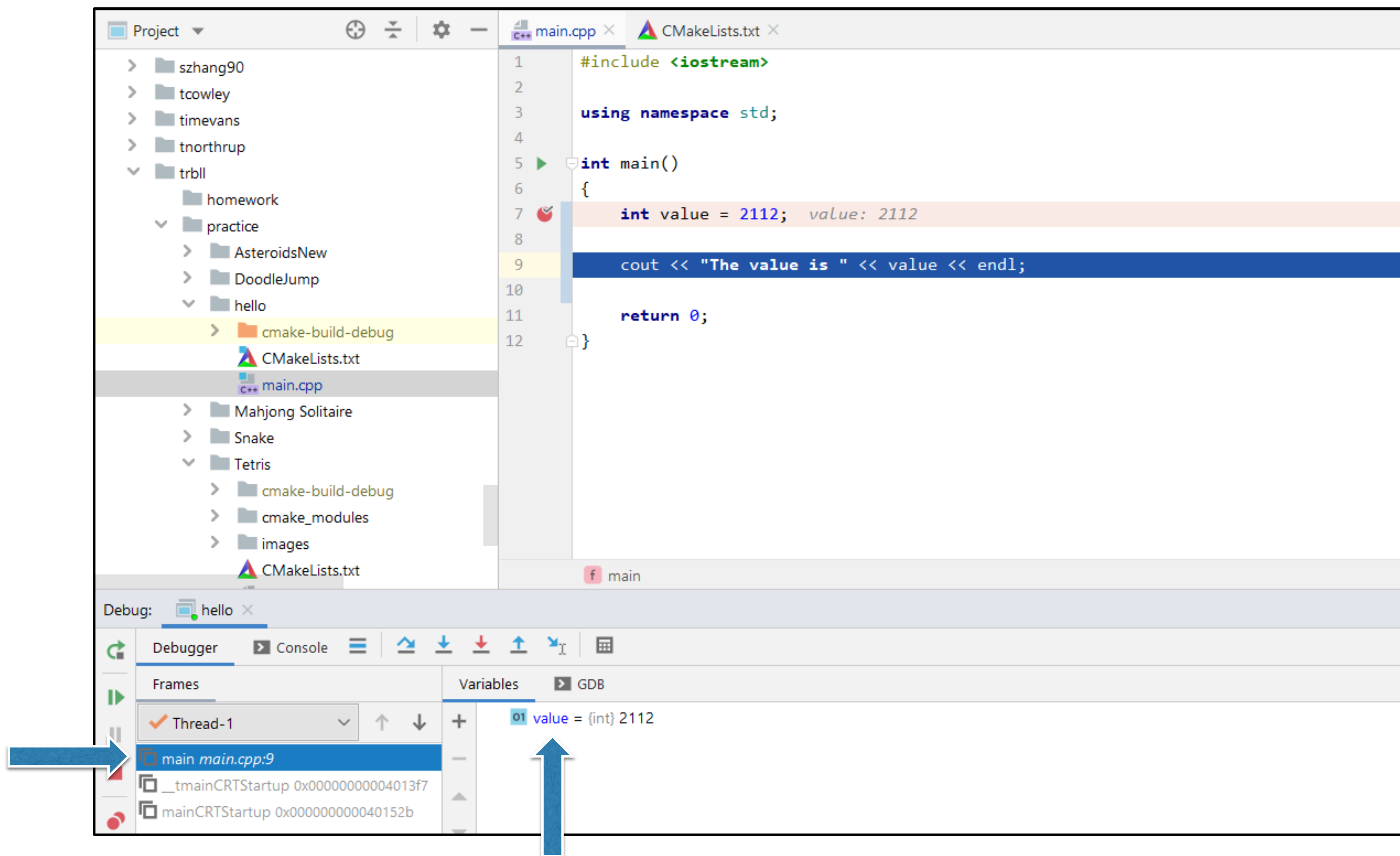
Step Out
of Function



Step Into
Function on Line



CLion Debugger Demo





CQ: What is printed as a result of calling the following function?

```
int main()
{
    float a_f = 9.0;
    float b_f = 10.0;
    int a_i = 9;
    int b_i = 10;

    cout << a_f / b_f << ", " << a_i / b_i << ", " << b_i % a_i << endl;
    return 0;
}
```

integer division: $9/10 = 0$ (truncated)

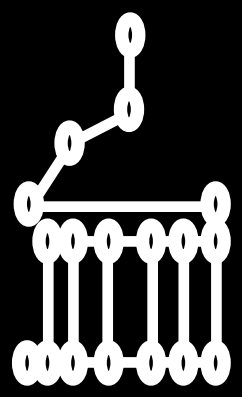
remainder after
dividing 10 by 9 = 1

A. 0.900000, 0, 0

B. 0.900000, 1, 0

C. 0.900000, 0, 1

D. 0.900000, 1, 1



Write a program to compute the average of 5 numbers

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout << "This program computes the average of five integers." << endl;
```

```
    cout << "Please enter five integers: ";
```

```
    int x1, x2, x3, x4, x5;
```

```
    cin >> x1 >> x2 >> x3 >> x4 >> x5;
```

```
    cout << "avg = " << (x1+x2+x3+x4+x5) / 5.0 << endl;
```

```
    return 0;
```

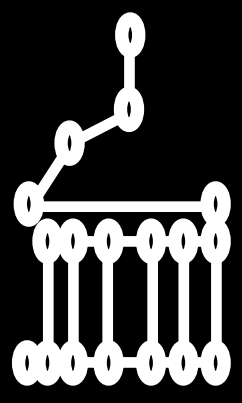
```
}
```

Question: Why divide by 5.0?

Answer: To prevent integer division.

Question: How would you extend this program to average 100 numbers?

Answer: Use a for-loop.



Write a program that computes the average of five numbers using a function.

```
double average5(double x1, double x2, double x3, double x4, double x5);
```

```
int main()
{
    cout << "This program computes the average of five numbers." << endl;
    cout << "Please enter five integers: ";
    int a1, a2, a3, a4, a5;
    cin >> a1 >> a2 >> a3 >> a4 >> a5;

    cout << "avg = " << average5(a1,a2,a3,a4,a5) << endl;
    cout << "avg = " << average5(1,1,1,1,2) << endl;

    return 0;
}
```

↓ Pass-by-value, i.e., copy input values into these variables.

```
double average5(double x1, double x2, double x3, double x4, double x5)
{
    return (x1 + x2 + x3 + x4 + x5) / 5.0;
}
```



Write a program to compute the average of 100 numbers

```
int main()
{
    cout << "This program computes the average of 100 integers." << endl;
    cout << "Please enter 100 integers: ";
    int x, sum = 0;
    for(int i = 0; i < 100; i++)
    {
        cin >> x;
        sum += x;
    }

    cout << "avg = " << sum / 100.0 << endl;

    return 0;
}
```

Important: Don't forget to set the sum to zero. Why?
Answer: otherwise, sum will have an unknown value.

Question: Does it matter if you use ++i or i++?
Answer: No.

Question: How would you enter 100 numbers?
Answer: Use a file.