

## **Problem Set 0**

### **Setting Up the C/C++ Build Process**

Release date: 8<sup>th</sup> August 2020

**Due: N/A**

## **Information**

The objective of this problem set is to guide you to setup a build process that you can use to write, compile and execute C/C++ programs. Also, we include a simple exercise to help you familiarize yourself with the basics of C/C++, and also learn how to submit work through the Coursemology platform.

## **Choosing Your Tools**

To write and run a C/C++ program, you will need to install two items:

1. An editor to write the program
2. A C/C++ compiler to compile the written C/C++ code into machine code for the computer to run.

This guide will walk you through installing these two items.

## **Step 1: Installing the Editor**

C/C++ programs are nothing more than text files, which can be written using any text editing software, like Notepad. More advanced editors provide more features like syntax colouring, bracket matching, auto-completion, etc. It is akin to using Notepad vs MS Word. Notepad might be sufficient for writing a short memo but it would be better to use MS Word if you are going to write a hundred page novel.

Software developers will use an Integrated Development Environment (IDE) to write code. But that will be quite an overkill for this class. Microsoft has recently released a lightweight editor called **Visual Studio Code**. We recommend you to use this editor if you have no other preference of your own. (*Note: this is different from Visual Studio, which is a full-fledged IDE by Microsoft*)

## **Installing Visual Studio Code**

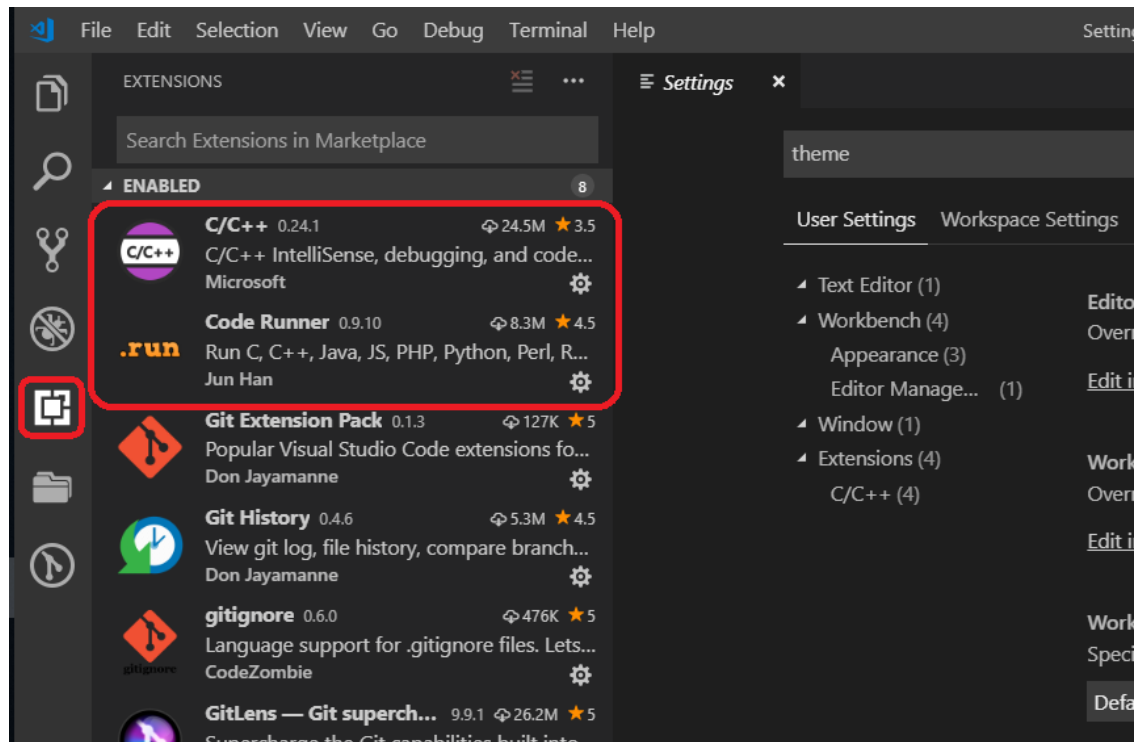
You can download Visual Studio Code from their website: <https://code.visualstudio.com/Download>. Platform specific guides for installing on Windows, Mac and even Linux can be found here: <https://code.visualstudio.com/docs/setup/setup-overview>

## Visual Studio Code Extensions

Visual Studio Code is a generic code editor. It requires extensions to be installed to work with C code.

To install extensions, simply click on the **Extension** icon on the left toolbar, or press `Ctrl+Shift+X` (on Mac `⌘+⇧+X`). You can then type in the name to search for in the marketplace.

We recommend installing the following two extensions:



1. **C/C++** by Microsoft

This gives several services like syntax highlighting, auto-completion, symbol searching, etc. which makes it easier to write C code.

2. **Code Runner** by Jun Han

Code Runner is a convenience extension that simply run a command-line instruction to compile and then execute the current file. Once installed, you can simply right-click anywhere in the editor and select **Run Code**, or press **Ctrl+Alt+N** to compile and run your code. Or you can click the `>` button on the upper right corner.

## Step 2: Installing the Compiler

Simply put, a compiler is a program that translates C code into machine readable code which the computer can execute. Just like text editors, there are several different compilers written for the C language, each having its own features. Well known ones include GNU C Compiler, Clang and Microsoft Visual Studio. Different compilers may recognise different “dialects” of C, i.e., what one compiler might treat as valid C code, another compiler might reject.

These differences will not matter much for our course, as we will only be dealing with basic C. We will adopt the GNU C Compiler (GCC) as the standard compiler for this class. You are, however, free to use whichever compiler on your own PC.

## Windows

There are three main options for those using the Windows operating system, listed in order of ease-of-installation:

1. MinGW
2. Cygwin
3. Windows Subsystem for Linux (WSL)

This guide will only instruct you on the installation of MinGW. You are free to explore other compilers and build environments on your own.

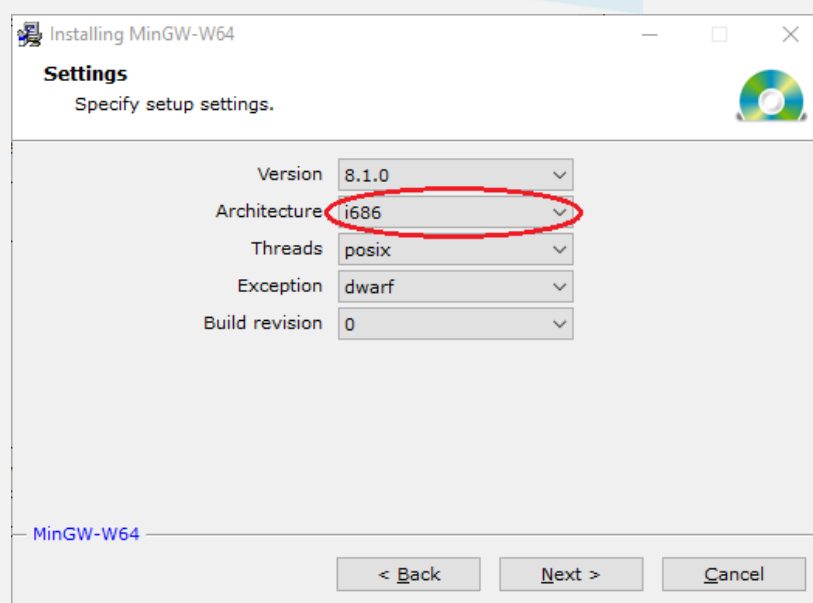
### Install MinGW

MinGW, a contraction of “Minimalist GNU for Windows”, is a minimalist development environment for native Microsoft Windows applications.<sup>a</sup> It is basically a Windows port of the GNU compiler tools like GCC, Make, Bash, etc. It does not attempt to be compatible with Unix so some features unique to Unix will not be available. Thus, the programs have to be specially written to run on Windows. However, as mentioned earlier, we will not require any of those feature in our class.

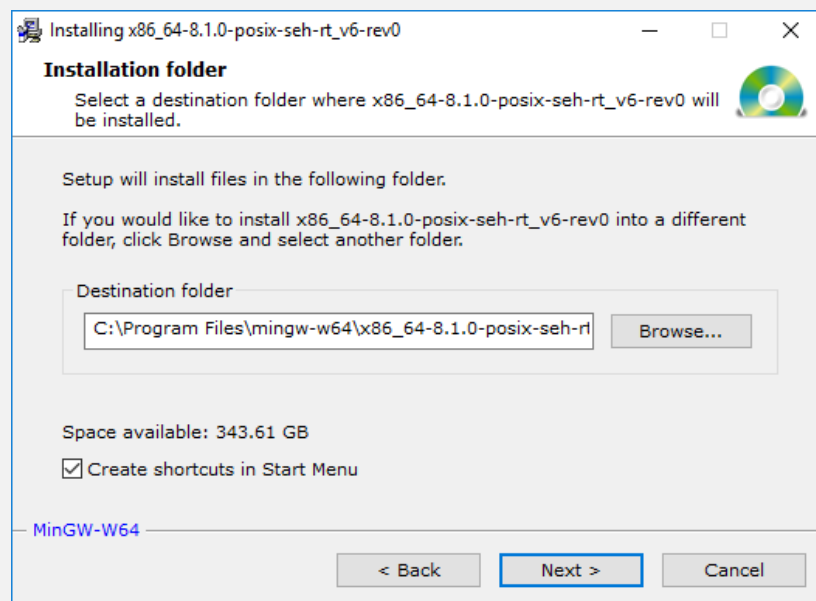
We recommend using an advancement of the original MinGW project, Mingw-w64. You can find more details on <https://mingw-w64.org> or directly download the installer from <https://sourceforge.net/projects/mingw-w64/>.

Run the downloaded file **mingw-w64-install.exe** to install.

When prompted to choose the settings, choose the correct architecture that matches your Windows<sup>b</sup>: i686 for 32-bit Windows, x86\_64 for 64-bit Windows.



Next you will be prompted for the installation folder. We recommend you use the default, though you may change it if you wish. In any case, take note of the destination folder.



### ***Adding to path***

After installation, it is best to add the binary files to your PATH so that you can run the compiler from any directory in your terminal. The following instructions will guide you through this.

*For Windows 10:*

1. Open **User Account** from the **Control Panel**, or by typing it in the Start Menu.
2. Select **Change my environment variables** on the left side pane.
3. Under **User variables for XXX**, select **PATH** and click **Edit**.
4. Click **New** to create a new entry and type something to add it. Then click on this new entry and select **Browse** and browse to the **bin** folder of where Mingw-w64 is installed. By default it should be in *C:\Program Files\mingw-w64\x86\_64-... \mingw64\bin*.
5. Finally, due to some compatibility with PowerShell, you will need to remove the surrounding quotes, if any. Simply double-click on the field and delete the quotes.

*For Windows 7:*

1. From the desktop, right-click **My Computer** and click **Properties**.
2. In the System Properties window, click on the **Advanced** tab
3. In the Advanced section, click the **Environment Variables** button.

4. Finally, in the Environment Variables window, highlight the **Path** variable in the User Variables section and click the **Edit** button. Each different directory is separated with a semicolon.
5. Append the full directory path to the **bin** folder of where Mingw-w64 is installed. By default it should be in *C:\Program Files\mingw-w64\x86\_64-...\mingw64\bin*.

### Verifying the Installation

Open a Command prompt (from the Windows Start Menu) and enter “**g++ --version**”. If the installation was *unsuccessful*, you will see something like:

'g++' is not recognized as an internal or external command,  
operable program or batch file.

If you see something like:

```
g++.exe (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Congratulations, you have successfully installed the compiler. GCC is directly accessible from your Windows Terminal.

*Note: Executable files compiled with MinGW runs natively in Windows. As such, your antivirus software might perform real-time scanning when executing the program. This might delay the start by several seconds. Disabling real-time scanning or setting some exclusion in your antivirus can remove this delay.*

<sup>a</sup>MinGw. <http://mingw.org>

<sup>b</sup><https://support.microsoft.com/en-sg/help/827218/how-to-determine-whether-a-computer-is-running-a-32-bit-v>

## Mac OS

Before trying to install GCC, first check if you already have a C compiler installed. The easiest way is to open a Terminal window and enter “**g++ --version**” and see if the command is recognized.

If you get a command not found like this:

```
-bash: g++: command not found
```

it means you do not have a compiler installed and will have to install GCC.

### Installing GCC

An easy way to install and manage applications on Mac OS is to use the Homebrew package manager.

1. Install Homebrew by entering the following in the Terminal prompt as one line:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Do not copy and paste from the PDF as it will insert a line break. Alternatively, you can visit <https://brew.sh/> and copy and paste the line shown.



2. Enter the following line in the Terminal prompt:

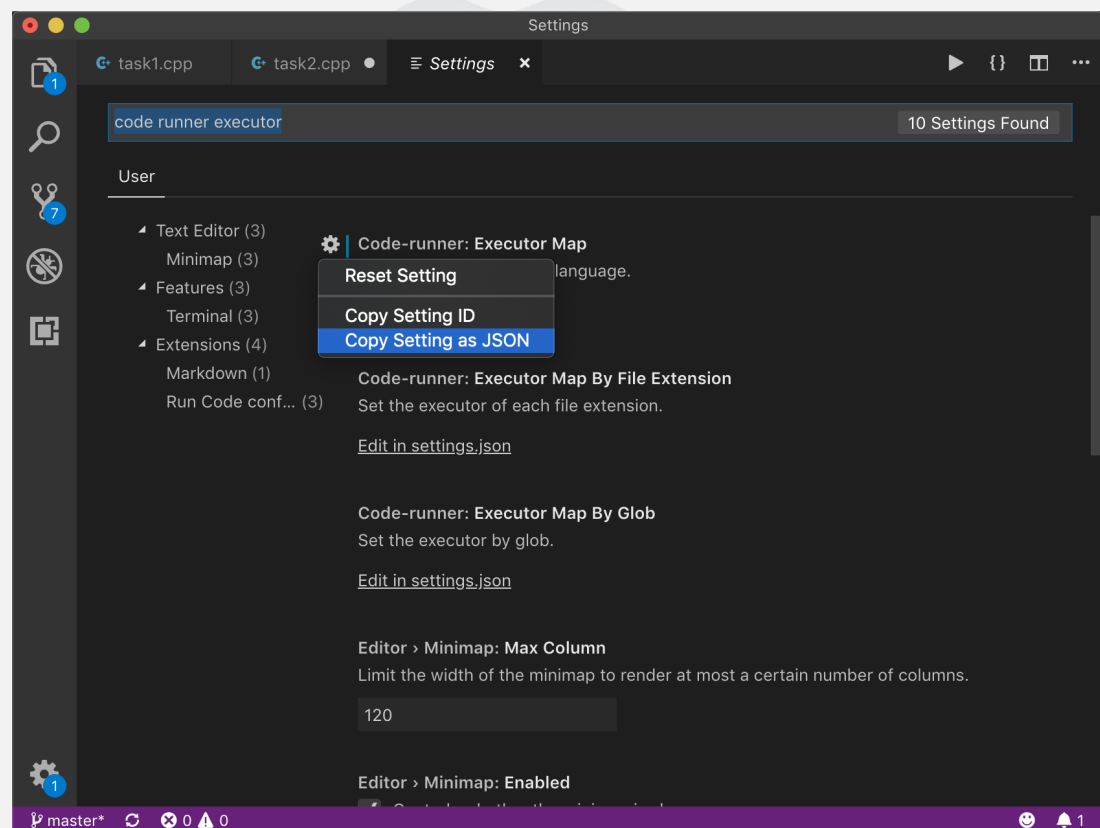
```
brew install gcc
```

3. Once done, run “**g++ --version**” to verify the installation. You should see a line with something like Apple LLVM version 9.0.0 (clang-900.0.39.2). The exact numbers do not matter, as long as you do not get “command not found”.

## Customize Visual Studio Code

Unfortunately, the default C/C++ version that the Mac compiler uses is not C++11. We will have to configure VSC to use C++11, otherwise some of your programs will not compile.

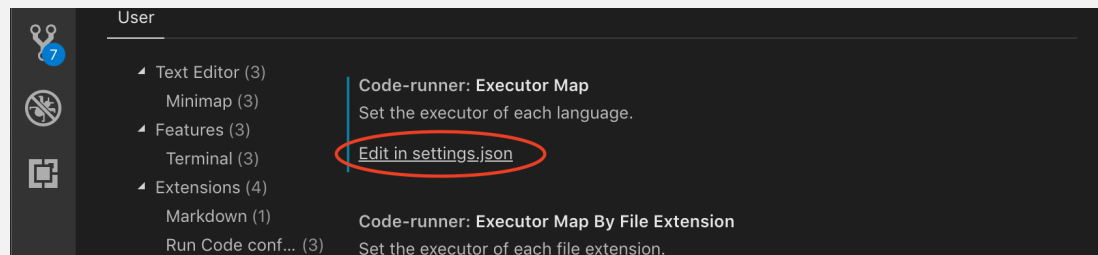
1. In Visual Studio Code, open the **Settings** pane by pressing  + .
2. In the search bar, type “**code runner executor**”. A list of settings matching the search term should appear.



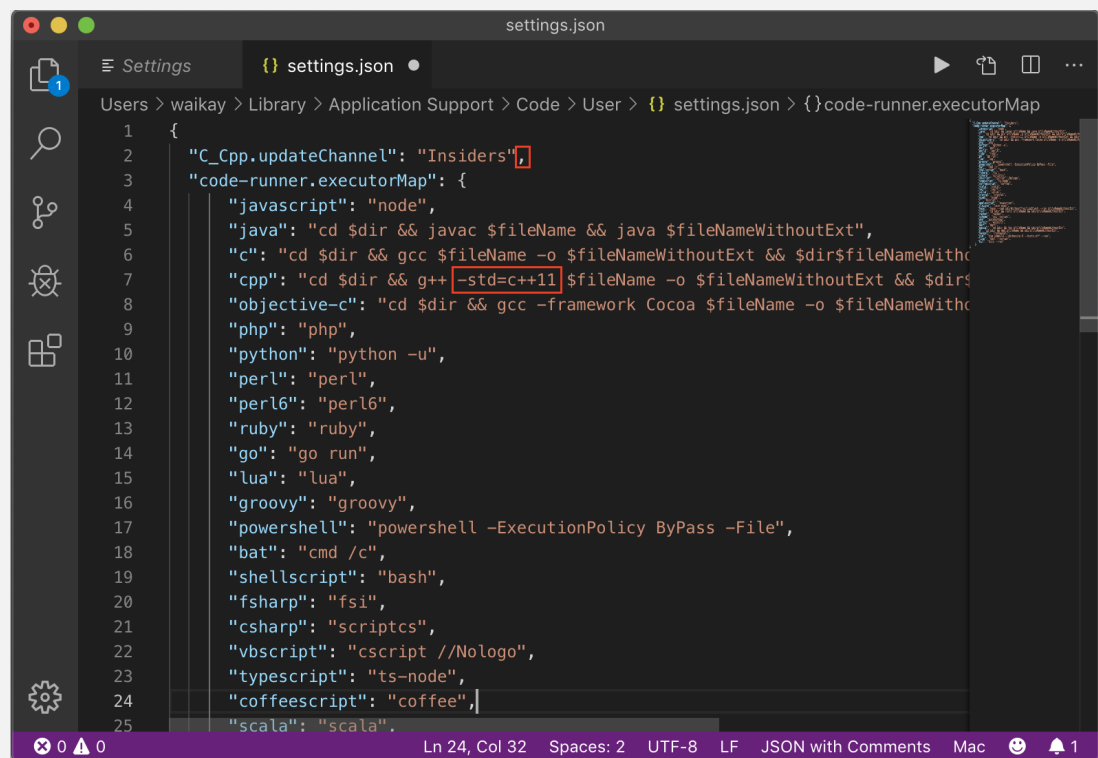
Mouse over the empty space next to **Code-runner: Executor Map** and a gear icon should appear.

Click on the gear icon and select **Copy Setting as JSON**

3. Now click **Edit in settings.json** under the **Code-runner: Executor Map**




4. A new tab should open you should see a list of options as shown:



If not, paste the contents from the clipboard between the braces as in the screenshot above.

If there are existing lines, put a **comma** at the end of the last line and paste the contents after that.

5. Find the line beginning with **"cpp"** and add the text **-std=c++11** between **g++** and **\$filename** as shown above.

6. Save the file by pressing  + 

## Step 3: Testing Your Setup

To test if you have set up the build environment correctly, we will write a test program, compile and execute it.

1. Create a new file in Visual Studio Code..
2. Type in the following lines:

```
#if __cplusplus <= 199711L
#error You require a C++11 compliant compiler
#endif

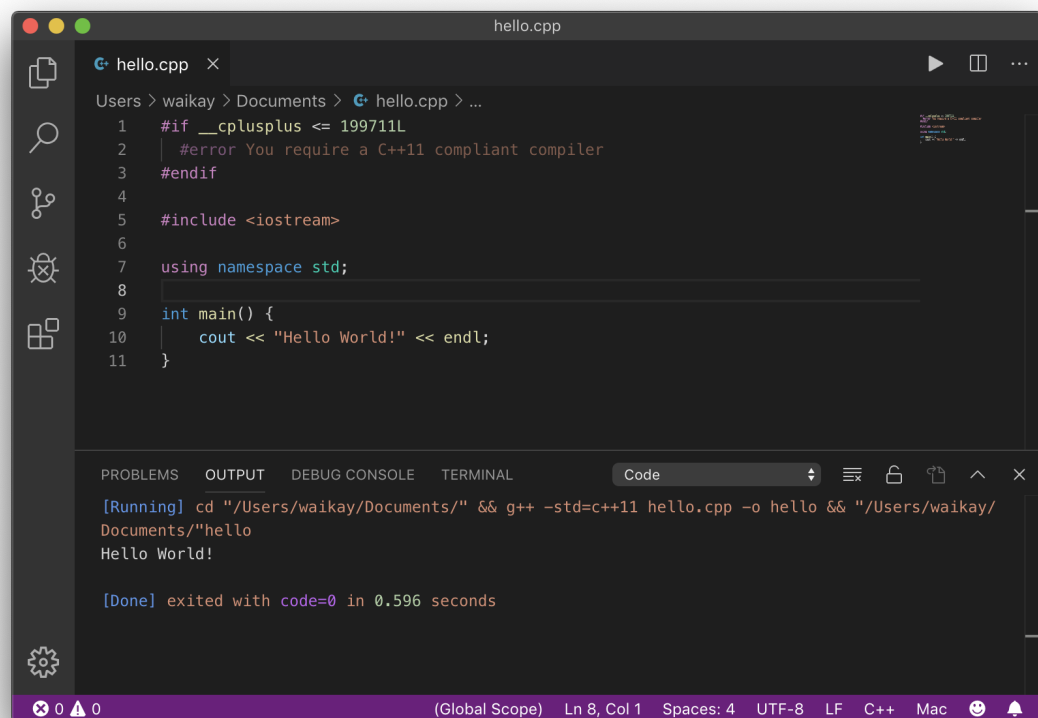
#include <iostream>

using namespace std;

int main() {
    cout << "Hello World!" << endl;
}
```

3. Save the file as **hello.cpp**
4. Run the file by clicking on the ▶ at the top-right corner, or press **Ctrl** + **Alt** + **N** (on Mac: **ctrl** + **⌘** + **N**).
5. A black region should appear at the bottom of the window and you should eventually see the words **Hello World!** displayed, like so:





The screenshot shows a code editor window titled 'hello.cpp'. The code is as follows:

```
1  #if __cplusplus <= 199711L
2  | #error You require a C++11 compliant compiler
3  #endif
4
5  #include <iostream>
6
7  using namespace std;
8
9  int main() {
10 |     cout << "Hello World!" << endl;
11 }
```

Below the code editor is a terminal window with the following output:

```
[Running] cd "/Users/waikay/Documents/" && g++ -std=c++11 hello.cpp -o hello && "/Users/waikay/
Documents/"hello
Hello World!

[Done] exited with code=0 in 0.596 seconds
```

The status bar at the bottom indicates the file is at line 8, column 1, with 4 spaces, UTF-8 encoding, LF line endings, and is a C++ file on a Mac.

6. Your set up works! Grab a beer and give yourself a pat on the back.

### [Optional] Information on compiling your code

Compiling your code is done in a terminal/command prompt with the following command: `g++ --std=c11 <prog.cpp>` where `<prog.cpp>` is the name of your program file.

The compiler will create an executable file called `a.out` (or `a.exe` on Windows). You can then run the file by simply entering `./a.out` (or `a.exe` for Windows).

## Submitting on Coursemology

Most of the programming tasks require you to define a function. You only need to copy the required functions and paste into the Coursemology code window. There is no need to copy and paste the `main` function. The `main` function is found in the downloaded template to help you test your program locally on your PC.

More will be explained in Lab 1.