

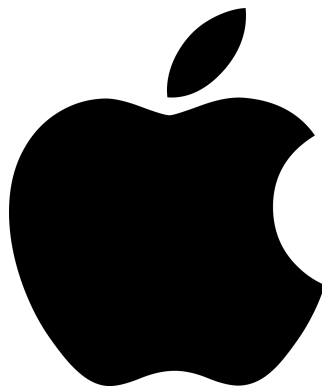
# Important Things to Consider for E115

## Intro to Programming

### 2020 edition, Mac Users

updated, August 20, 2020

Brandon Seidman  
E 115  
Stevens Institute of Technology



# Xcode

(Special thanks to Ethan Dy Tioco)

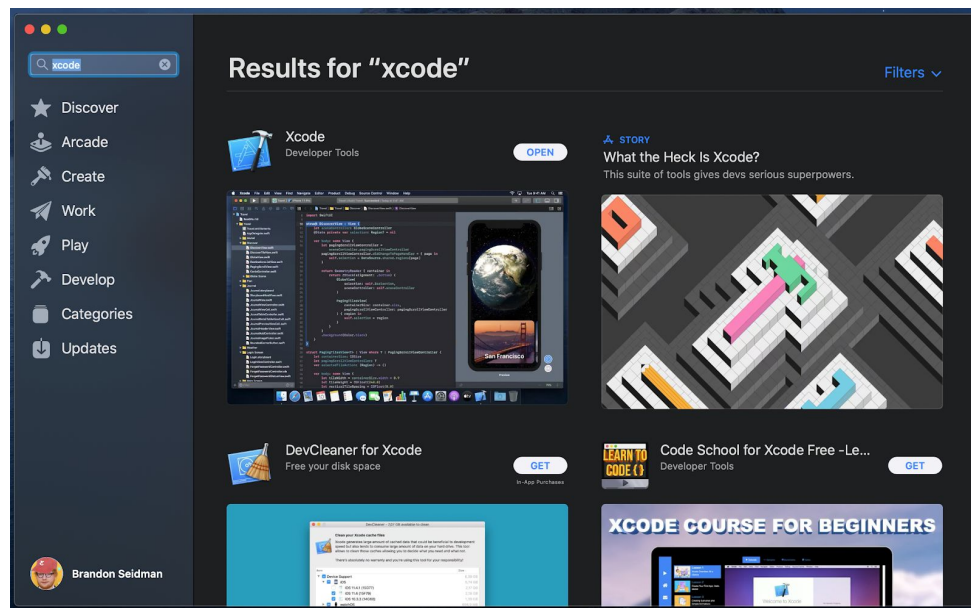
## **Table of Contents**

<b>2</b>	<b>..... The Easy Path - Xcode</b>
2	..... Installing Xcode
3	..... Setting Up Xcode
5	..... Typing Your .cpp Program
<b>7</b>	<b>..... The Hard Path - Terminal</b>
7	..... Getting Started
8	..... Setting Up a .cpp Program
9	..... Locating Your Program in the Terminal
<b>12</b>	<b>..... Running Your Program in the Terminal</b>
<b>14</b>	<b>..... Handing in Class Material</b>
14	..... Homeworks
15	..... Projects
16	..... Tests
<b>17</b>	<b>..... Teaching Assistants (TA's)</b>
<b>18</b>	<b>..... References</b>

## The Easy Path - Xcode:

### Installing Xcode:

The language we will be using in this class for all of our assignments will be C++. To run C++ you have a couple different options on Mac. Xcode is an app by Apple and is the simplest option, though you will soon see why it can be a bit limiting. For the scope of this class at least though, Xcode will be perfect for what we will be doing. Installing Xcode is super simple, all you have to do is **go to the App Store and download it from there.**

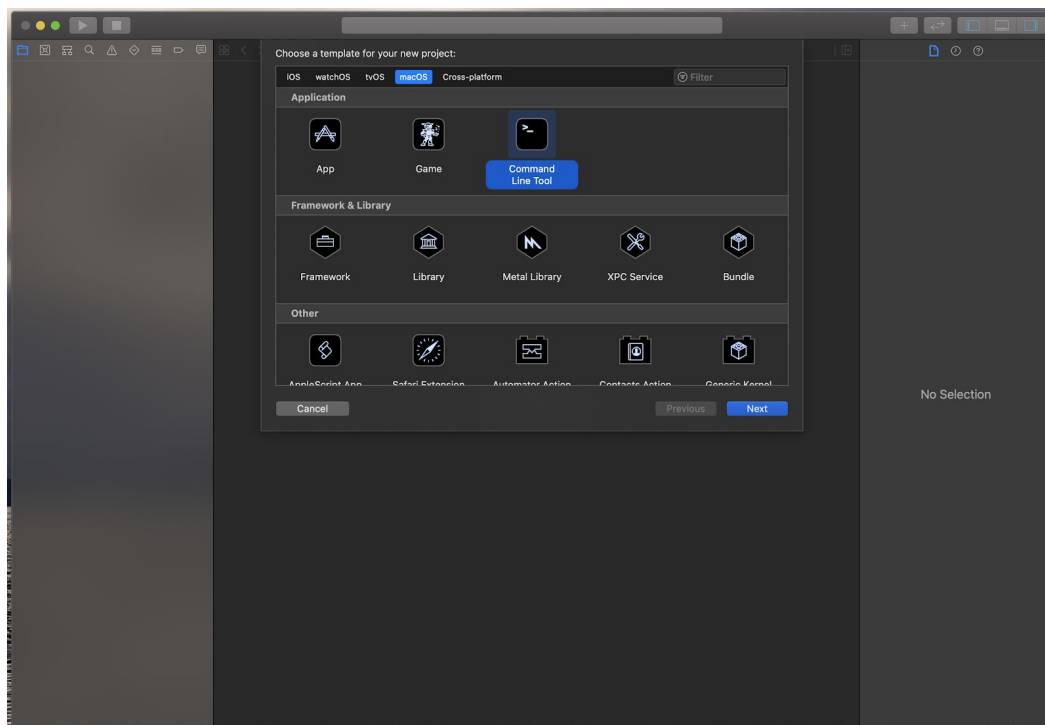


## Setting Up Xcode:

Once you have downloaded Xcode you will be brought to this screen upon opening it. When you are going to create an assignment you will first pree **“Create a new Xcode Project,”** but if you have already created it and just need to edit it it will appear on the right side of the screen.



Next you will have to choose the type of project you want to create. For C++ we will be choosing **“Command Line Tool”** on the **“macOS”** tab.



Next we will choose all the settings for our project. **The product name can just be the assignment name (ex: “Homework1”), the organization name and Identifier can just be your name, and most importantly we need to make the language C++.** Everything else can be ignored.

Choose options for your new project:

Product Name: Homework1

Team: Add account...

Organization Name: Brandon Seidman

Organization Identifier: Brandon Seidman

Bundle Identifier: Brandon-Seidman.Homework1

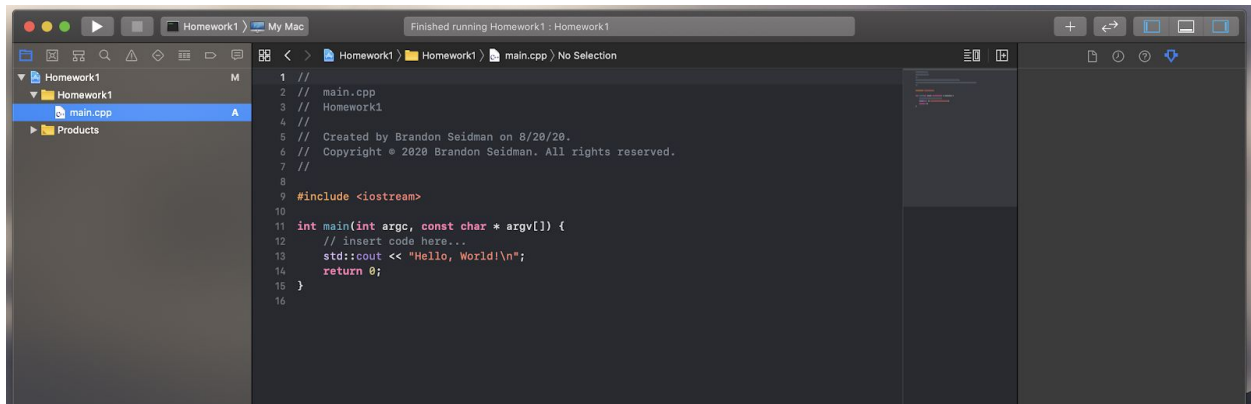
Language: C++

Cancel Previous Next

After this you can save the project to wherever it's convenient but it's preferable that it's in a place that you know because you may need to access the files later.

## Typing Your .cpp Program:

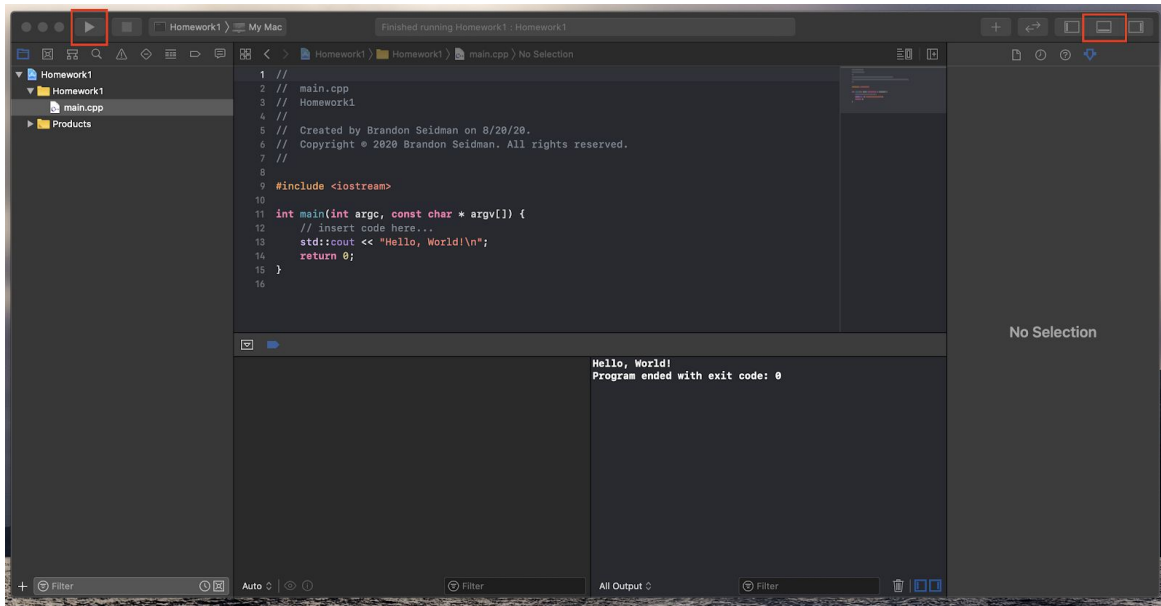
Great so now you're all set up and ready to code! Once you're into your project you should see a file on the left side called **"main.cpp."** **This is where all your actual coding will be done** and when you set it up you should already have code there for a hello world program. This is perfect because our next step will be to test and make sure everything is working. If it's not however, below the picture I have written out a simpler version of this code that we will probably go over the first couple days of classes. Just copy and paste that into your "main.cpp" file and move on to the next step to start testing.



```
#include <iostream>  
using namespace std;
```

```
int main(){  
    cout << "Hello, World!" << endl;  
    return 0;  
}
```

To test the code it's super simple in Xcode all you have to do is press **that little play button in the top left**. If you can't see the code you probably don't have the console open. If you **press the middle button in the top right corner**, it will open up the console where your code will output. If the console says "Hello, World!" then congratulations! You are now a programmer. I've attached your certificate below but please only print it if you actually got this working, otherwise contact your local TA and bother them until they fix it or contact me because no one knows how Macs work. Now if you want your next reward take the hard path as well, or just skip it if you don't like awards.



## The Hard Path - Terminal:

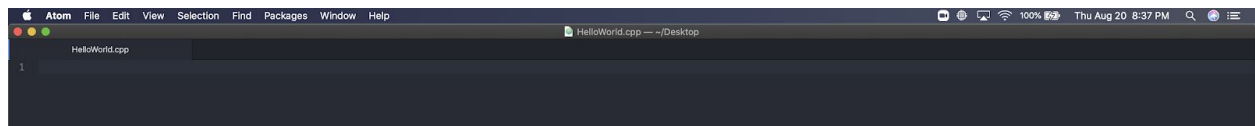
### **Getting Started:**

If you're here you're probably here because you either like a challenge, have programmed before, or heard through the grapevine that this is the only way to get homework 1 to print in color if you are a Mac user. No matter what the reason you can still use Xcode and follow all the steps for the easy way and still use terminal or you can use any IDE of your choice. **Popular choices include Visual Studios (very versatile), Sublime (very simple), Atom (my personal choice and great if you use Github), VIM (very hard but super useful, TextEdit (technically you can but don't), or any other text editor of your choice.** I can't go through installing and setting up all of them but for the most part all of them just involve downloading, typing, and saving as a cpp file. I will do this in Atom below as an example but if you have problems with any other there are plenty of tutorials online and the TAs should be able to help you as well. This section is more about being able to run cpp files in Terminal though, which is very powerful and super useful knowledge if you plan on pursuing other programming classes or programming as a career.



### Setting Up a .cpp Program:

No matter what IDE you choose to use, your first step is to **create a new file with the “.cpp” extension**. Below I have created a new file in Atom and named it “HelloWorld.cpp.” Make sure that you save this file where you can easily find it. I have folders in my Documents folder where I save all my projects for example. If you end up not being able to figure out where a file is located then these next couple steps are going to go from hard mode to impossible mode real quick. Your next step is to simply write code into your file. Below the image I have written out the code code for “Hello, World!” Simply copy and paste this for now then move onto the next step. (Also if the next step seems really hard make sure you read up until “The Right Click Method.”)



```
#include <iostream>
using namespace std;

int main(){
    cout << "Hello, World!" << endl;
    return 0;
}
```

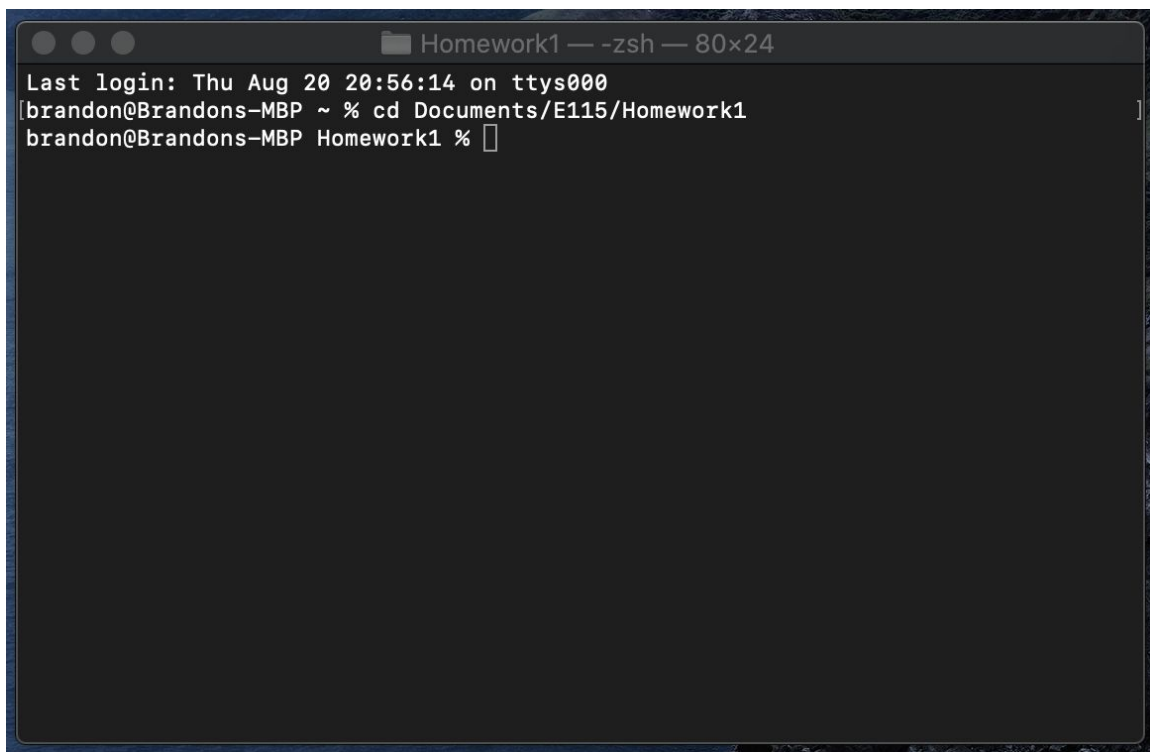
## Locating Your Program in the Terminal:

For this step you are going to start to use your first Unix commands in the terminal. This is super powerful and it's what makes Macs so powerful when it comes to programming. So next time your friend tells you Macs are worse than PCs simply explain to them the powers Unix grants to you for 2 hours, this will ensure they never talk to you about PCs nor anything else ever again!

When using Terminal (pre-installed on all Macs) to find a file your most powerful tools are "cd" and "ls." "cd" stands for change directory and when you follow it with a path to a folder it will bring the terminal to that folder. Terminal will always start at the base user directory and cd will bring it to any directory you want. For example if my cpp file was in the "Documents -> E115 -> Homework1" folder, I would type this command into terminal:

```
cd Documents/E115/Homework1
```

Lets see what happens when I do this:

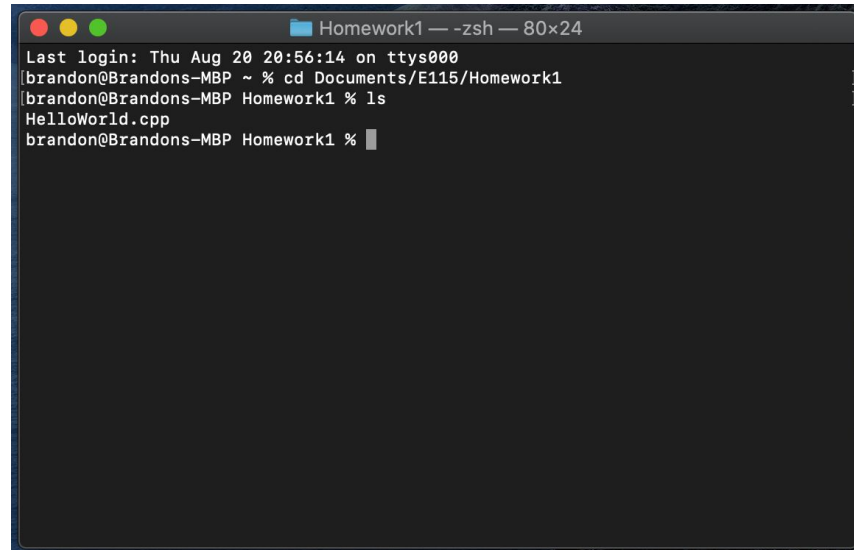
A screenshot of a macOS Terminal window. The title bar at the top reads "Homework1 — zsh — 80x24". The terminal content shows the following text: "Last login: Thu Aug 20 20:56:14 on ttys000", followed by a prompt "brandon@Brandons-MBP ~ %". The user has entered the command "cd Documents/E115/Homework1", and the prompt has changed to "brandon@Brandons-MBP Homework1 %". A cursor is visible at the end of the new prompt.

```
Last login: Thu Aug 20 20:56:14 on ttys000
brandon@Brandons-MBP ~ % cd Documents/E115/Homework1
brandon@Brandons-MBP Homework1 %
```

You can see in the photo that the "~," which represents the home folder I was initially in, now says "Homework1," the folder I am currently in. This might be tough at first and you might mess things up, but that's alright! Below I have written out 5 surefire ways to make this much easier:

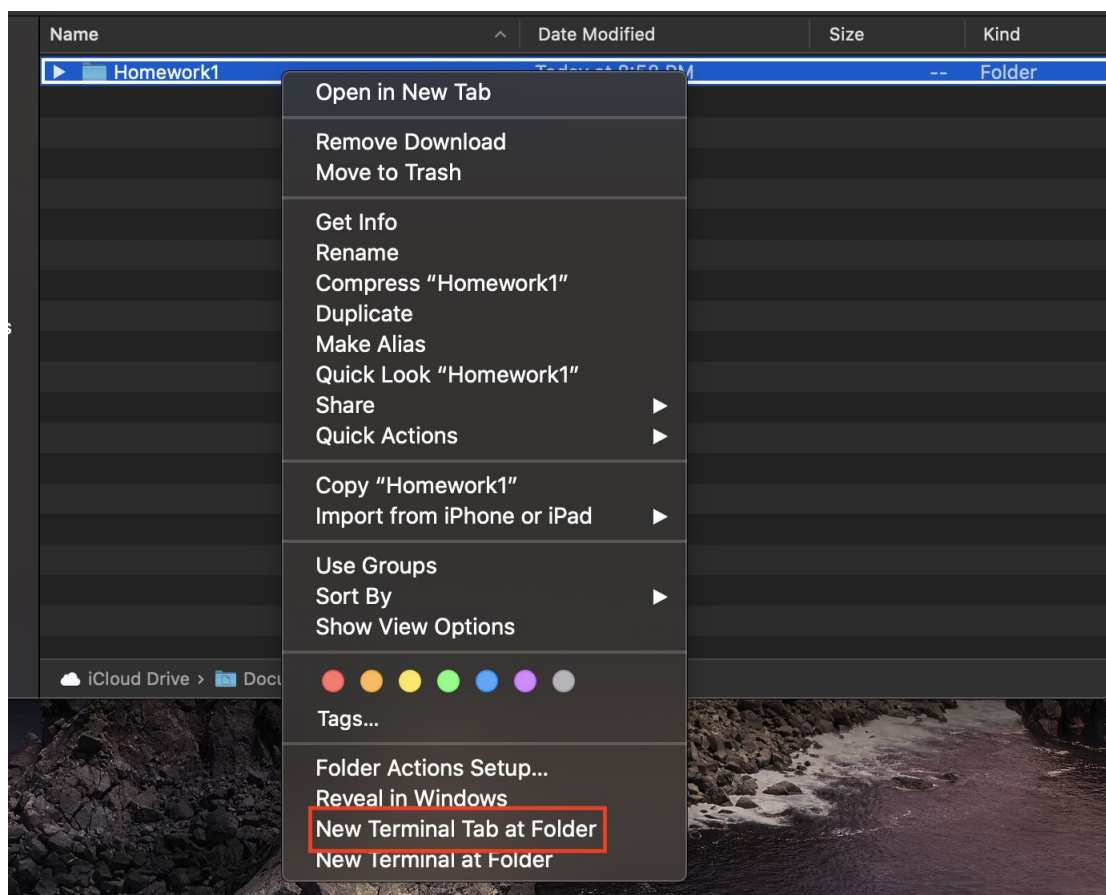
1. ls - First and foremost, that is an L, not an I. "ls" is another command like "cd" that you can use in Terminal. What it does is it lists out everything that is contained in your

current directory. This is super helpful because with this we can always check that we are in the right folder and what the name of the next folder is. Below is a picture of me using the “ls” command to check and make sure that my cpp file is in fact in the Homework1 folder:

A terminal window titled "Homework1 — -zsh — 80x24". The terminal shows the following commands and output:

```
Last login: Thu Aug 20 20:56:14 on ttys000
brandon@Brandons-MBP ~ % cd Documents/E115/Homework1
brandon@Brandons-MBP Homework1 % ls
HelloWorld.cpp
brandon@Brandons-MBP Homework1 %
```

2. Use Tab - Tab is your best friend when looking through folders. If you type in the first couple letters of the file you're looking for and press tab it will automatically fill in the rest for you. No more typos!
3. Separate Commands - In my example I wrote “cd Documents/E115/Homework1” but what if I can’t remember all those folders! What if I want to use ls to check if I went to the right folder! Simple, just do these all separately. First “cd Documents” then “cd E115” and finally “cd Homework1”. This gives you the same exact result with much less room for mistakes.
4. cd .. - It’s very easy to get way ahead of yourself, what happens if you went to the wrong folder or a folder too far? In this case “cd ..” is your new command-z. This will bring you back 1 space so if I “cd ..” from my Homework1 folder I will now be in the E115 folder.
5. The Right Click Method - This is all very hard and new, I get it. Now normally, I wouldn’t condone cheating but this is our way to cheat the system a bit. Eliminate everything you just learned from your mind and simply right click on the folder where your cpp file is contained. On the dropdown you will see an option that says “New Terminal Tab at Folder.” Just press this and boom! No more Unix commands, just good old right click. Check out the picture below for an example:

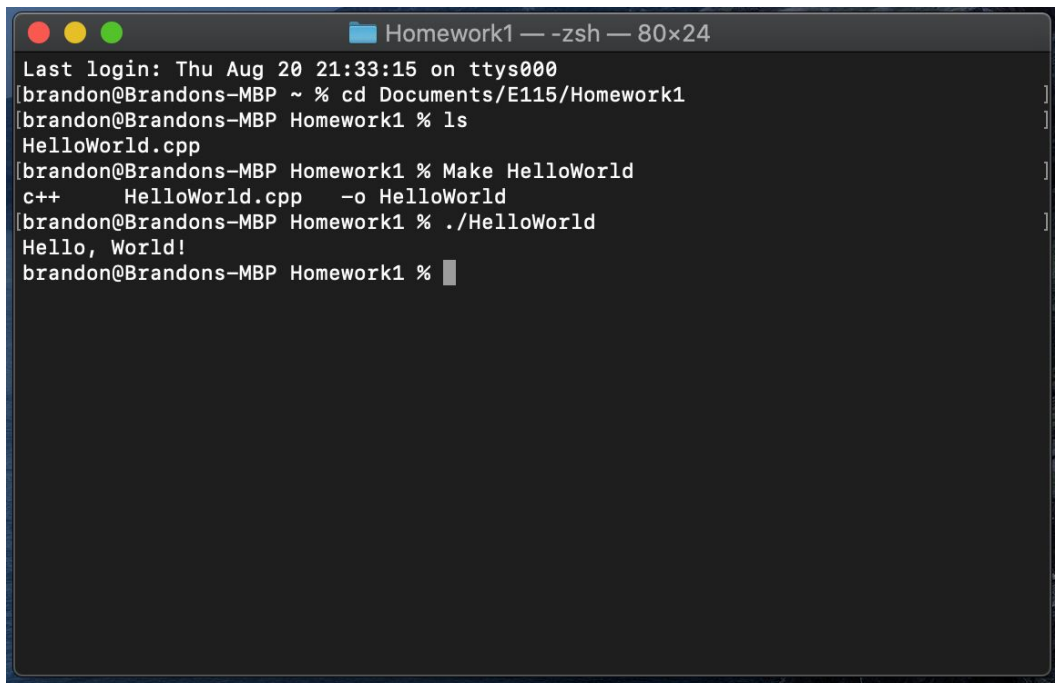


### Running Your Program in the Terminal:

Once you've located your cpp file the next logical step is to run it. Once again we are going to need unix commands and **we will be using 2 simple ones, "make" and "./" to run the program.** Make is going to set up your file to run and ./ will actually run it. If my file is named HelloWorld.cpp here are the two commands I will run in terminal:

```
make HelloWorld
./HelloWorld
```

You can use "ls" to make sure you are in the right directory by checking to see if your file is there. Also when you use this method it will create another file similar to a .exe for windows. When submitting assignments this isn't important to include but it needs to be there to use the ./ command. Make sure you run the make command as well whenever you make changes to the .cpp file. Below is an image with an example of what this all will look like:

A screenshot of a macOS terminal window titled "Homework1 — -zsh — 80x24". The terminal shows the following sequence of commands and output:

```
Last login: Thu Aug 20 21:33:15 on ttys000
[brandon@Brandons-MBP ~ % cd Documents/E115/Homework1
[brandon@Brandons-MBP Homework1 % ls
HelloWorld.cpp
[brandon@Brandons-MBP Homework1 % Make HelloWorld
c++      HelloWorld.cpp  -o HelloWorld
[brandon@Brandons-MBP Homework1 % ./HelloWorld
Hello, World!
brandon@Brandons-MBP Homework1 %
```

Once you have this all working you are now not only an official programmer but you also get your badge of Unix Mastery. Print that out, stick it on your certificate, staple it to your backpack, do whatever. Congrats.



## Handing in Class Material

### Homework:

For homework, you will need to submit the following:

1. **.cpp file**
2. **screenshots of the command window running the code (i.e. the output)**

To get the .cpp file if you are using Xcode, you have two solutions. The first is just memorizing where you create your project and going there and finding it (it will be named main.cpp). Or you can right click where it says "main.cpp" in Xcode and press "Show in Finder" to automatically go to that folder. If you use the hrd method just make sure you're saving the file somewhere you can easily find and make sure you know the name.

As for screenshots, Macs come with a pre-installed screenshot app. Just open that up and make sure your screenshot includes the coderunning in either Xcode or the Terminal and you are good to submit. Every once in a while the TAs ask for additional screenshots or info but if that happens we will let you know.

If you've missed the deadline, don't worry! Always strive to hand in your homework, even if it's late. Homework contributes to a *huge* part of your grade, so it's best to have a healthy partial credited homework, than no homework at all. Make sure to communicate with your TA's, and don't wait until the last week of the semester to hand everything in because we have grading deadlines and it's really not fun to grade every assignment during finals week.

## **Projects:**

**Hand in all essential files** that are included in your project on canvas. Many great projects can be condensed into one .cpp file, just like your homework. Some may require a .txt file for file streaming. Other projects can be full-blown games made in Unity, which would require either an installation file, or the .exe game file and a folder with all the game data. Whatever files you made, include them all. If you're handing in a folder, it's a good idea to make it a .zip file. Additionally when projects become very hard to figure out how to run TAs may ask for screenshots, videos, or even small presentations of them working so just make sure your keeping up contact especially if you do something really huge and impressive.

**Each group member should individually hand in their group project on Canvas.** Groups can be made across different sections, which is good! However, if only one member hands it in for the whole group, it will be hard for TA's to track down which student did what project. Make sure you also comment in your code or on Canvas who you worked with.

**HAND IN THE PROJECTS ON TIME.** You'll read later that TA's have to follow strict grading deadlines. Please spare them the heartache of giving you a zero or a heavily-penalized grade because you did not submit your project on time.



**Tests:**

Tests are normally done through Google Forms. **HAND THESE IN ON TIME.** These forms close out and take a while to grade so if you are late with tests there's not much we can do. The tests are not that hard and can be done in groups, there is no reason you can't get these in on time so please please please don't miss a test. There are 2 a semester: a midterm and a final.

## **Teaching Assistants (TA's)**

Make sure that you have your TA's email addresses written down somewhere! Some of them are comfortable with sharing their phone numbers as well. TA-student communication is very important, and in many cases, *very time sensitive*. Keep a close eye out for emails or announcements from them.

Don't be afraid to ask the TA's for help! The average student for E 115 more likely does not have prior coding experience, so the class material can seem intimidating. The TA's are there to help and guide you to understand the fundamental programming concepts that are covered in this class. They *want* to help you out. So, whether it's a homework question, or a lecture topic question, or even a technical question (like navigating through Visual Studios), ask away and ask quickly!

If you have any grading questions as well, make sure to contact your TA's immediately. They have grading deadlines to follow, and it would be easier to sort things out sooner than later.

More than anything, we are students just like you. We understand how much work you have and what you are going through. Please use us as a resource as much as possible. This class will only be a struggle if you make it one, help us help you.

## **References**

[1] Important Things to Consider for E 115 Intro to Programming 2019 edition, Windows Users