CSCI 1300 CS1: Starting Computing
Ashraf, Cox, Spring 2020
Homework 0
Due: Saturday, January 18, by 6:00 pm
(5 % bonus on the total score if submitted by 11:59 pm January 18)

Zip file submission must be completed and submitted by Saturday, January 18 by 6pm for your homework to receive points.

---

1. Objectives

- Set up your Moodle account
- Apply for an AWS educate account
- Set up AWS Cloud 9 IDE
- Set up VS code
- Introduction to Linux
- Submit a zip file with HelloWorld.cpp, HelloVSCode.cpp, and profile.txt to Moodle.

---

2. Tasks

2.1 Create Moodle Account:

All students in CSCI 1300 this semester will be accessing course materials through the Computer Science Moodle: http://moodle.cs.colorado.edu. To use Moodle, you log in with your identikey and password, and then enroll in your class.

Once you've logged in, click on the link to your class:
CSCI 1300 - Ashraf/Cox - CS1 Starting Computing

The enrollment key for CSCI 1300 is:
CSCI1300Spring2020

Once you're enrolled in the class, you should see the course materials that have been uploaded so far, organized by topic/textbook_chapter/week.

2.2 Set up AWS educate account and Cloud 9 IDE:

In recitation today, you will be getting started with AWS Cloud9, a cloud-based development environment, which is designed to provide a consistent development environment for all students. Follow the steps below. If you need help ask your TA or one of the CAs.

There are three phases for setting-up your AWS Cloud 9 IDE:

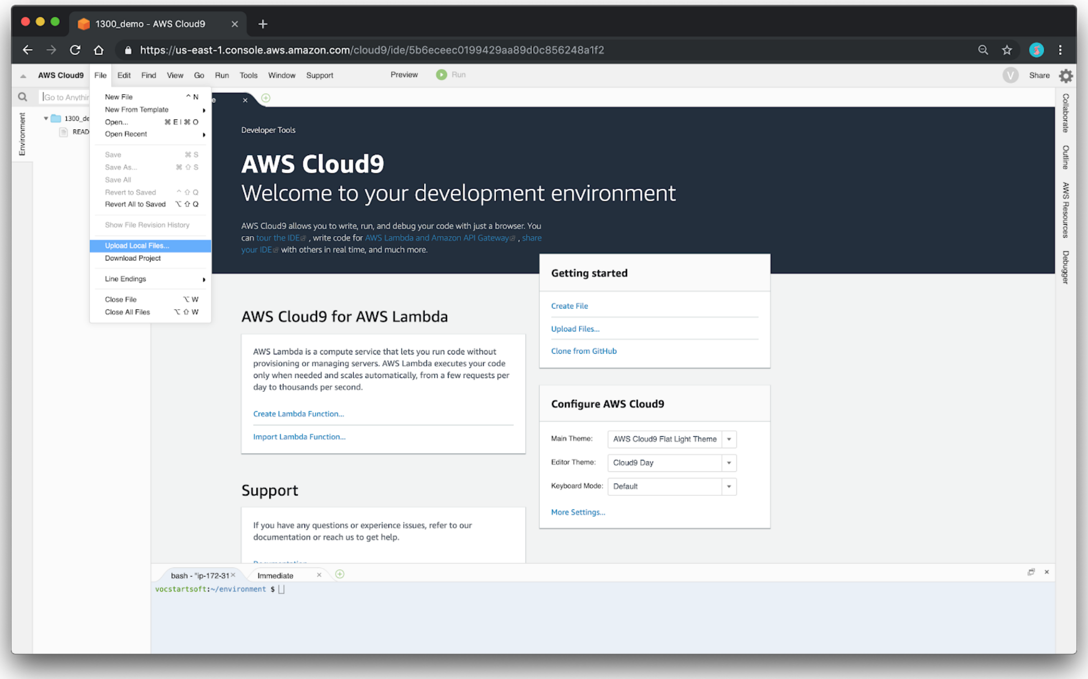
1. Apply for an AWS Educate account
2. Log-in to the student portal and go to the AWS classroom
3. Set up AWS Cloud 9 workspace

You can find [AWS cloud 9 set up guide]() on Moodle

2.3 Upload a profile.txt to AWS Cloud 9 IDE:

You can find Profile.txt file found on the first Moodle topic ("Introduction to Computers and Programming" – Topic 1) Download the profile.txt and upload it on your Cloud 9 workspace. Then, Insert your answers. Be sure to save your file!

1. From the menu bar on the top, choose **File** -> **Upload Local Files…**



AWS Cloud9

Welcome to your development environment

AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.

AWS Cloud9 for AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.

Configure AWS Cloud9

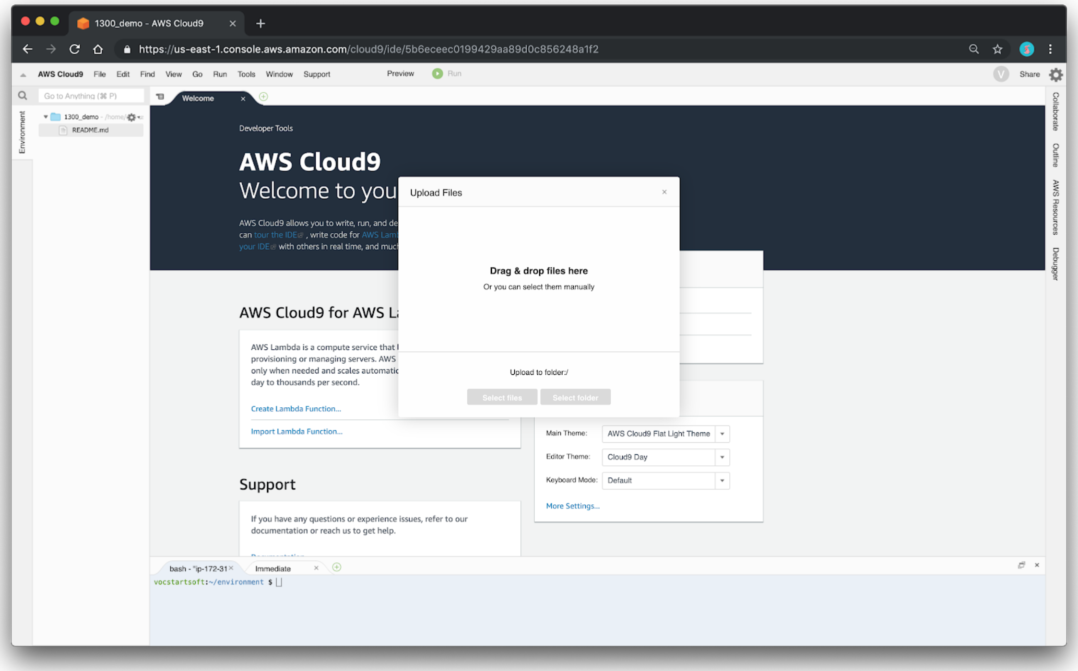
Main Theme: AWS Cloud9 Flat Light Theme
Editor Theme: Cloud9 Day
Keyboard Mode: Default

Support

If you have any questions or experience issues, refer to our documentation or reach us to get help.

```
vocstartsoft:~/environment $
```

2. Then, drag and drop the file you want to upload.


1300_demo - AWS Cloud9
https://us-east-1.console.aws.amazon.com/cloud9/ide/5b6eceec0199429aa89d0c856248a1f2
AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run
Share
Go to Anything (⌘ P)
Environment
1300_demo
README.md
Welcome
Developer Tools
Upload Files
Select files
Select folder
Create Lambda Function...
Import Lambda Function...
More Settings...
bash - "ip-172-31
Immediate
Collaborate Outline AWS Resources Debugger

AWS Cloud9

Welcome to you

AWS Cloud9 allows you to write, run, and de
can tour the IDE, write code for AWS Lam
your IDE with others in real time, and muc

Drag & drop files here

Or you can select them manually

Upload to folder:/

AWS Cloud9 for AWS La

AWS Lambda is a compute service that l
provisioning or managing servers. AWS
only when needed and scales automatic
day to thousands per second.

Main Theme: AWS Cloud9 Flat Light Theme

Editor Theme: Cloud9 Day

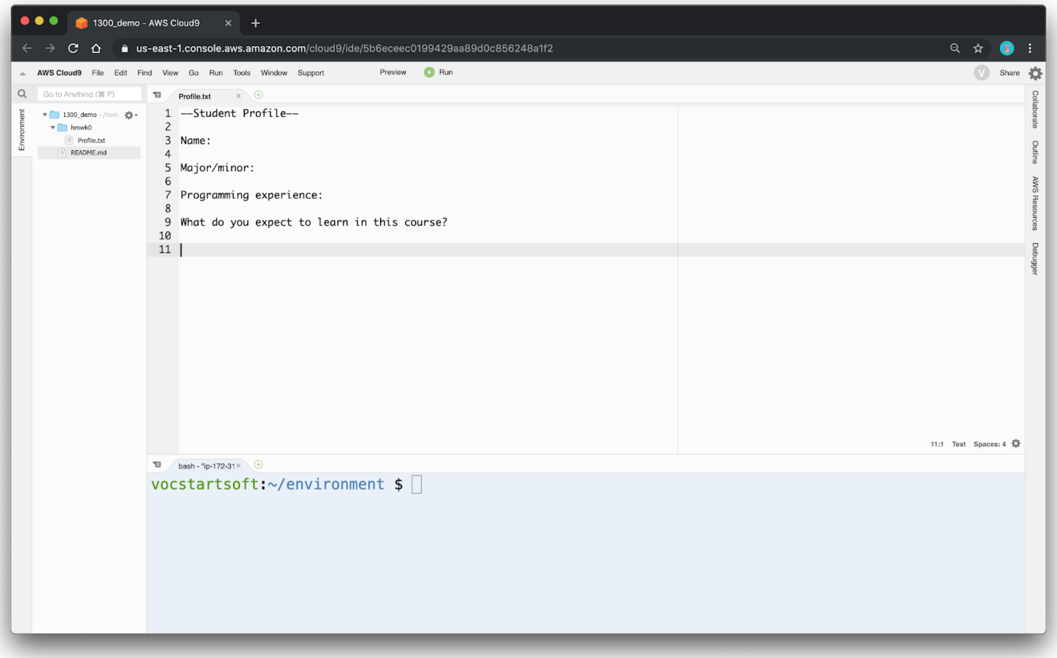
Keyboard Mode: Default

Support

If you have any questions or experience issues, refer to our documentation or reach us to get help.

```
vocstartsoft:~/environment $
```

3. Insert your answer and save it.


1300_demo - AWS Cloud9
us-east-1.console.aws.amazon.com/cloud9/ide/5b6eceec0199429aa89d0c856248a1f2
AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run
Share
Go to Anything (⌘ P)
Environment
1300_demo
hmwk0
Profile.txt
README.md
Profile.txt
11:1 Text Spaces: 4
bash - "ip-172-31
Collaborate Outline AWS Resources Debugger

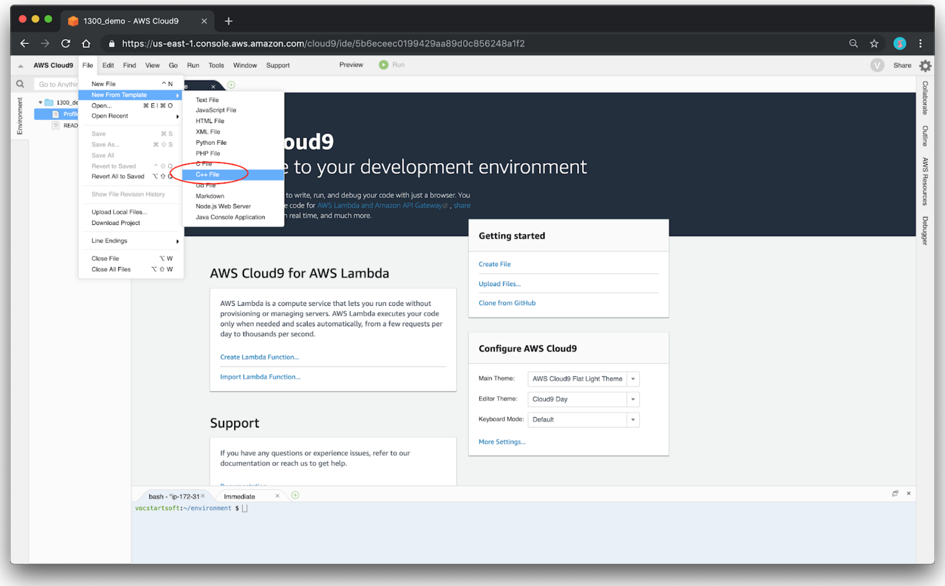
```
--Student Profile--

Name:

Major/minor:

Programming experience:

What do you expect to learn in this course?

```

```
vocstartsoft:~/environment $
```

2.4 Create a Hello World program:

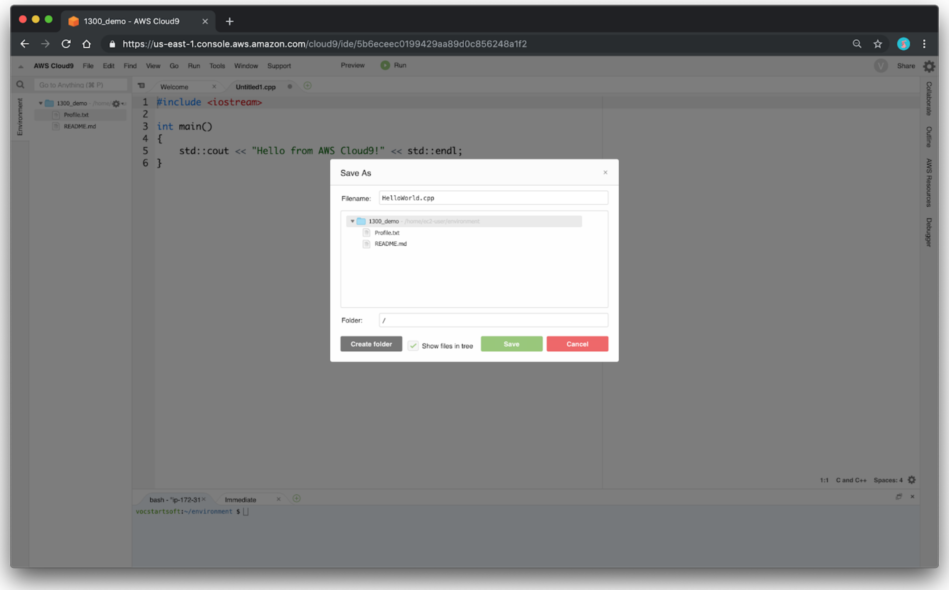
The "Hello, World!" program is one of the simplest programs in a programming language, and it is often used to illustrate the basic syntax of a programming language.

1. From the menu bar, choose **File** -> **New From Template** -> **C++ File**



```
vocstartsoft:~/environment $
```

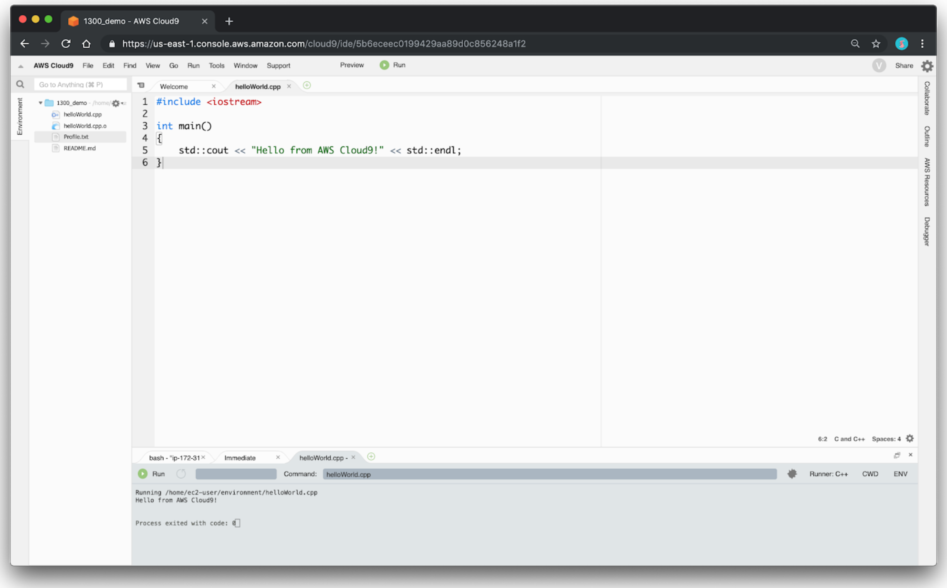
2. Save your file. **Important:** The filename should be named with **CamelCase** (the first letter of each word in a compound word is capitalized) end with **.cpp**. Your file name should be **helloWorld.cpp**



```cpp
#include <iostream>

int main()
{
    std::cout << "Hello from AWS Cloud9!" << std::endl;
}
```

```
vocstartsoft:~/environment $
```

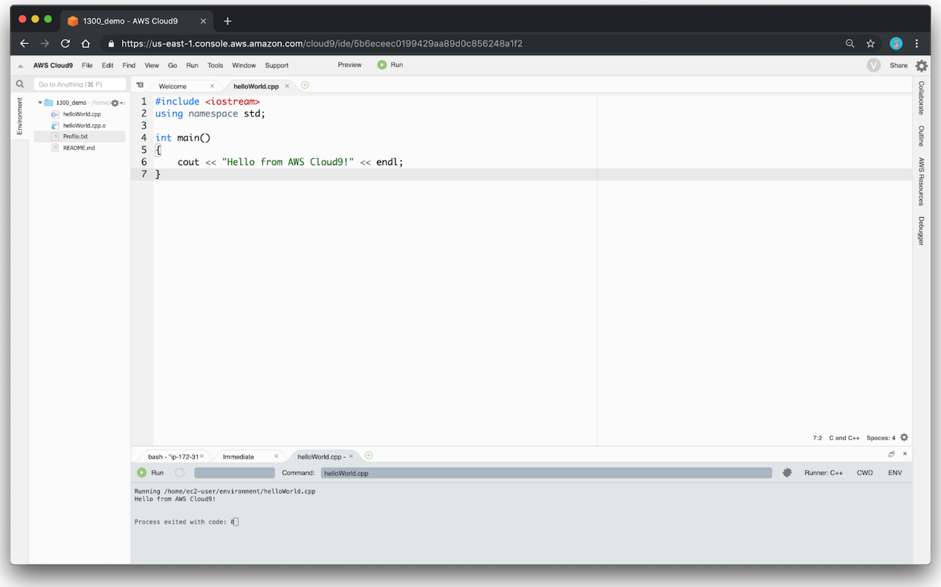
3. Click “Run” button on the top. You will see the output at the bottom of the page. The last line means that the program ended successfully.



```cpp
#include <iostream>

int main()
{
    std::cout << "Hello from AWS Cloud9!" << std::endl;
}
```

```
Running /home/ec2-user/environment/helloWorld.cpp
Hello from AWS Cloud9!


Process exited with code: 0
```

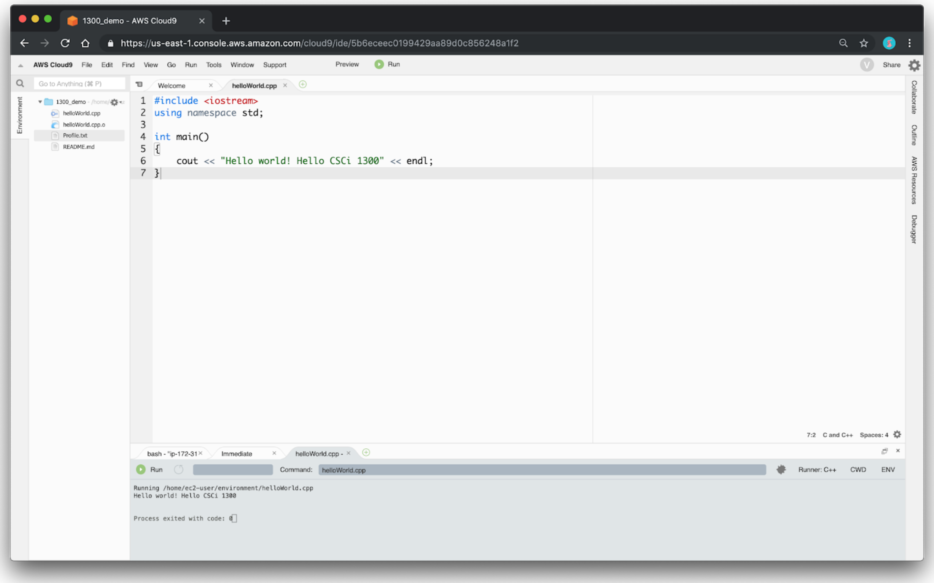
4. Let’s add a statement to use the *standard namespace*. Insert `using namespace std;` at the beginning of the code, then we can remove the std prefixes. Be sure to click “Run” button again to make sure that your code still compiles and runs.



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

int main()
{
    cout << "Hello from AWS Cloud9!" << endl;
}
```

```
Running /home/ec2-user/environment/helloWorld.cpp
Hello from AWS Cloud9!


Process exited with code: 0
```

5. Let’s modify the file to print “**Hello world! Hello CSCi 1300**”. The text inside of the quotation marks is printed as it is. It’s case sensitive too!

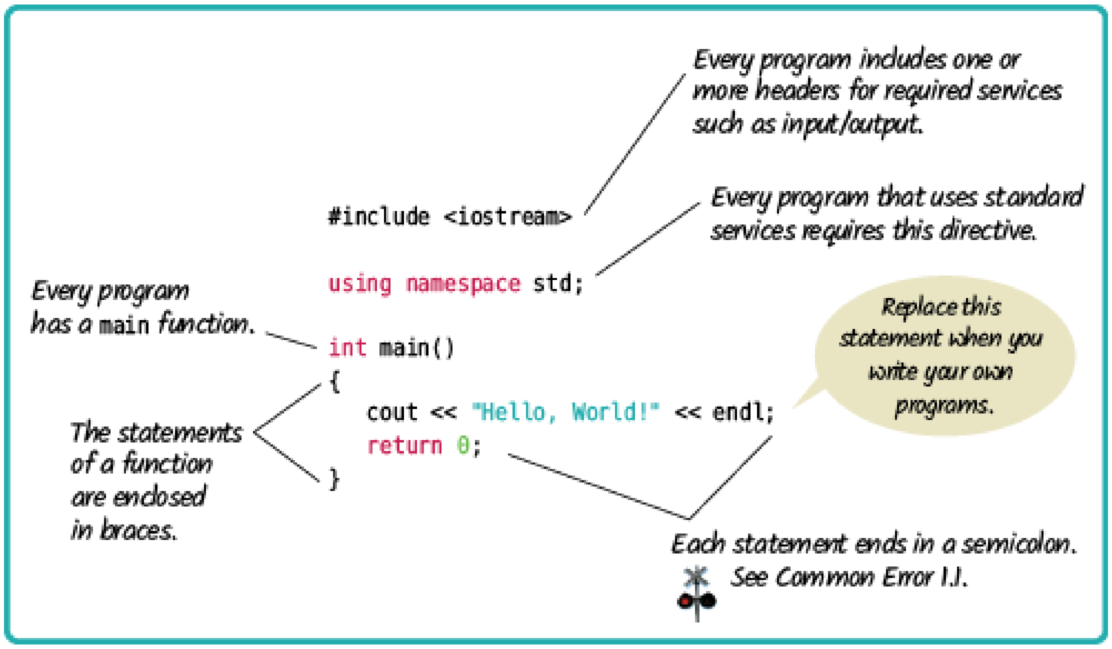


```cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world! Hello CSCi 1300" << endl;
}
```

```
Running /home/ec2-user/environment/helloWorld.cpp
Hello world! Hello CSCi 1300

Process exited with code: 0
```

Congratulations! You did the 1st C++ programming!

You will learn more C++ syntax over the semester to build your programming and problem-solving skills. Here is a snapshot of your program.



```cpp
#include <iostream>

using namespace std;

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

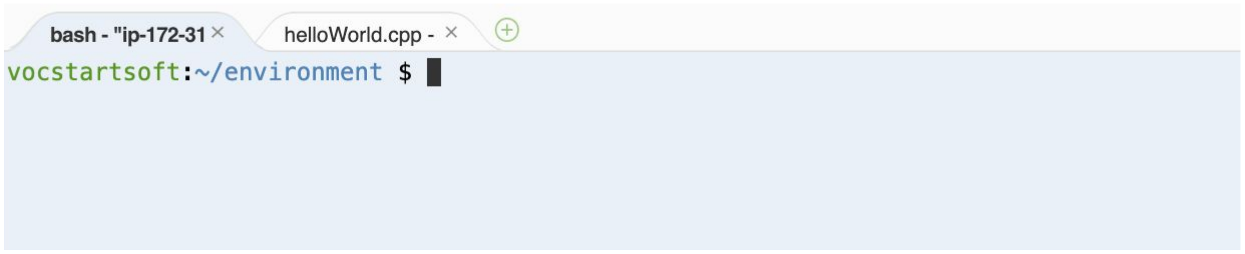
Every program includes one or more headers for required services such as input/output.
Every program that uses standard services requires this directive.
Every program has a main function.
Replace this statement when you write your own programs.
The statements of a function are enclosed in braces.
Each statement ends in a semicolon. See Common Error 1.1.

2.5 Learn how to use Terminal:

Switch to the *bash* tab and let's learn some useful Linux commands.

You see: `vocstartsoft:~/environment $`

- **~/** (i.e. everything else before the **$**) = your current directory (think of a directory as the folder you're in).
- **~** is shorthand for the current user's home directory.
- **$** represents the end of prompt for entering a command.



```
vocstartsoft:~/environment $ 
```

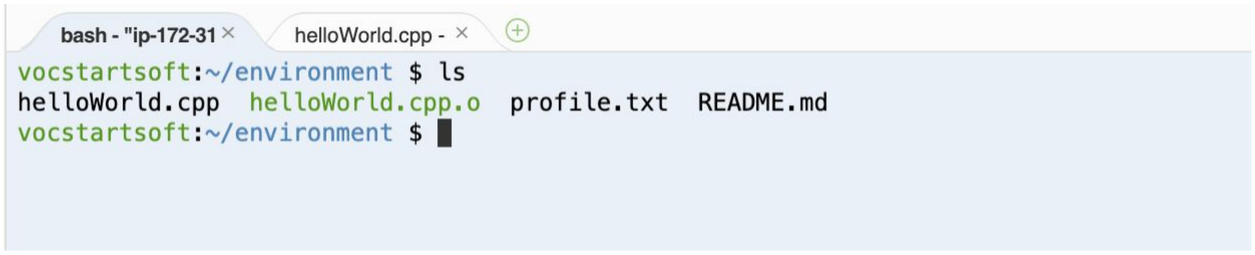
File browsing using the terminal is like using Windows explorer or like clicking on folders and navigating to different folders on your laptop. In the terminal, instead of clicking on folders we use commands to tell the computer what we want. If we want to go to a folder where we saved our last homework, we can type the commands to navigate to that folder and display its contents.

Try these commands:

`ls -- list directory contents`

`ls` = (that's a lowercase "L", not an uppercase "i") stands for list and is used to 'list' or show you everything in the current directory.

Note: after you type the command and press ENTER, something is displayed as a result of the command, but then the prompt returns.

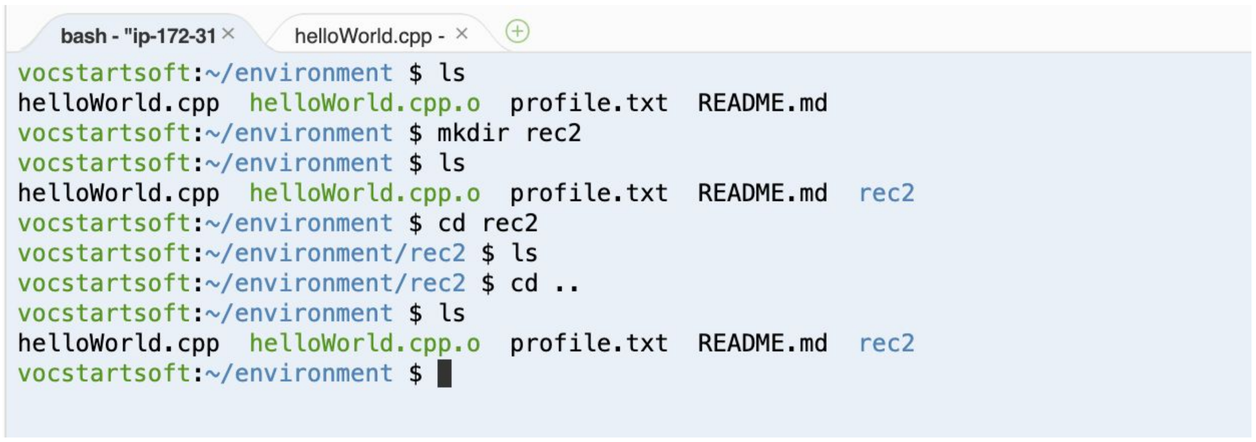


```
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md
vocstartsoft:~/environment $ 
```

`mkdir -- make directories`

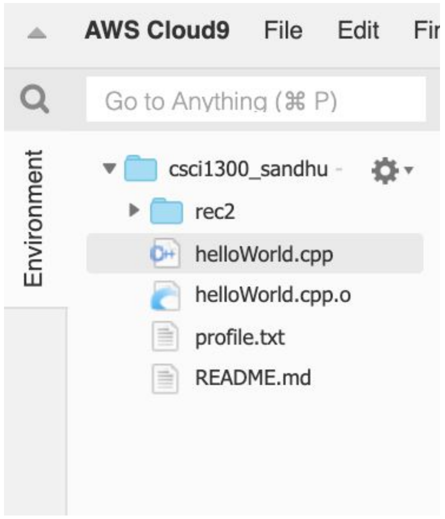
To create a new directory, use the command `mkdir <my_folder>`
Note: Spaces are troublesome on the command line, so we'll use underscores.

Let's create a directory for today's recitation, named `rec2`
`$ mkdir rec2`
`rec2` is now a directory/folder, nested under the folder environment. If you want to list the contents of the folder environment, you will now notice a new item: `rec2/`, where the forward-slash (`/`) indicates `rec2` is a subdirectory.



```
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md
vocstartsoft:~/environment $ mkdir rec2
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md  rec2
vocstartsoft:~/environment $ cd rec2
vocstartsoft:~/environment/rec2 $ ls
vocstartsoft:~/environment/rec2 $ cd ..
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md  rec2
vocstartsoft:~/environment $ 
```

You will also notice a new folder in the Environment tab, to the left of your window:



`cd -- Change the shell working directory.`

cd = stands for change directory is just like changing folders. It means, take me to place X. Commonly used as `cd <name_of_directory>`.

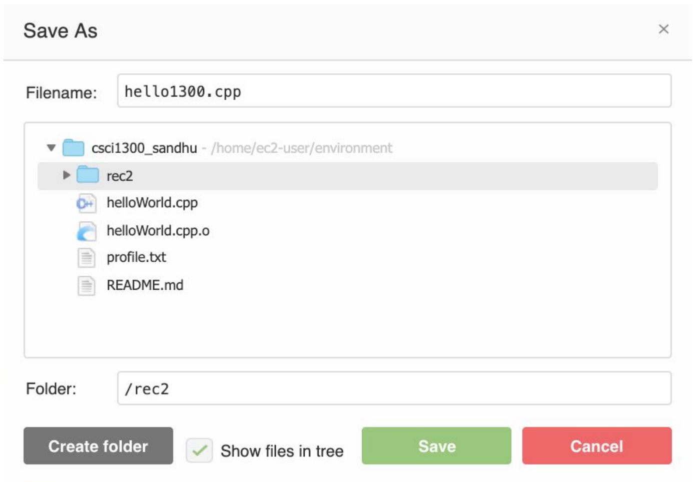
- **Note** that there will always be a space between cd and the name of the directory that you want to navigate to and the name of the directory will not include the carrots (side arrows) displayed above.
- **Note**: cd takes you places in reference to your current location. It's like going into a folder, and then clicking on a folder within that folder, and then clicking on another folder within that folder. You will always navigate deeper within that folder. To back up, we use "`cd ..`" (explained in a little bit).
- **Note**: cd (and other commands) is case sensitive. '`rec2`' does not equal '`Rec2`'. Make sure you type in directory names exactly as they are spelled.

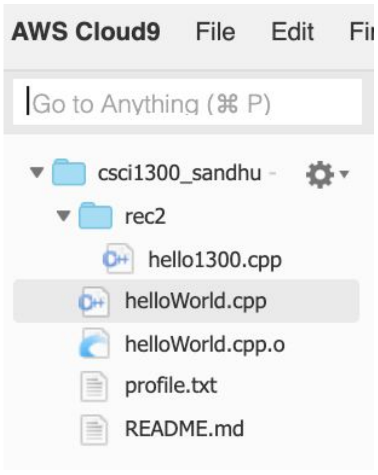
"`cd ..`" means go to the parent of my current location. It's essentially backing up.

cp -- copy files

(**Note**: Be very careful not to overwrite a file when copying.)

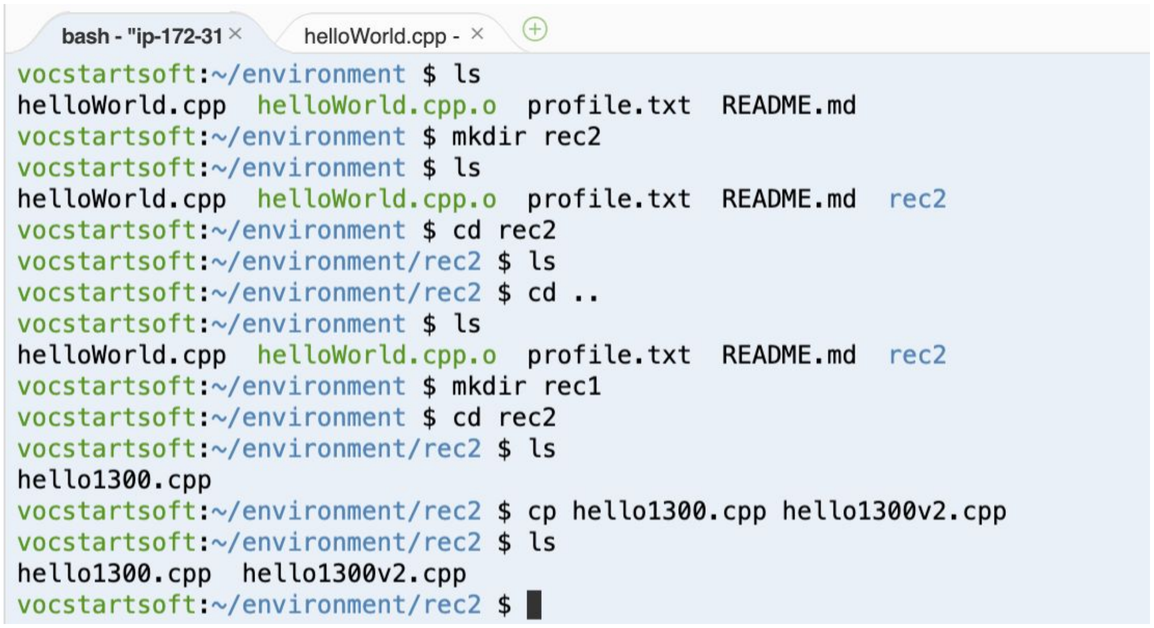
1. First, click inside the editor window, where the file `helloWorld.cpp` is open. Go up to the menu and choose File -> Save As and, when the dialog box opens, change the name of the file to `hello1300.cpp`. Choose `rec2` as the destination folder.





2. Once you click Save, the new file will appear in the file tree, in the Environment tab (see the figure on the right, above).
3. Let's make another directory named `rec1`.
   a. `$ mkdir rec1`
4. Now we want to copy the file from `rec2` into `rec1`. First, we go into the `rec2` directory:
   a. `$ cd rec2`
5. Ok, now we use the command for copying files:
   a. `$ cp hello1300.cpp hello1300v2.cpp`

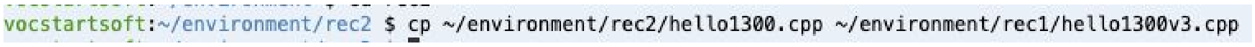
6. Now let's type ls and see what we have:



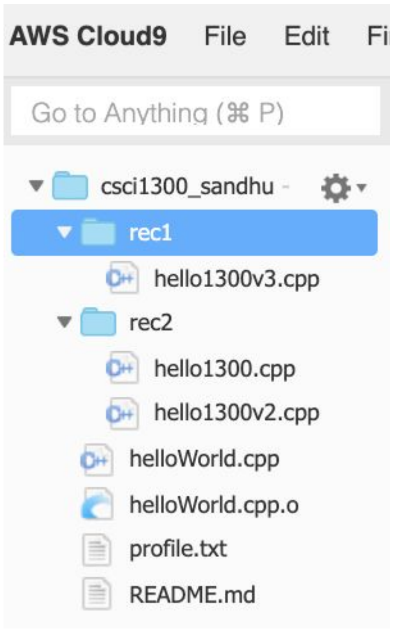
```
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md
vocstartsoft:~/environment $ mkdir rec2
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md  rec2
vocstartsoft:~/environment $ cd rec2
vocstartsoft:~/environment/rec2 $ ls
vocstartsoft:~/environment/rec2 $ cd ..
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md  rec2
vocstartsoft:~/environment $ mkdir rec1
vocstartsoft:~/environment $ cd rec2
vocstartsoft:~/environment/rec2 $ ls
hello1300.cpp
vocstartsoft:~/environment/rec2 $ cp hello1300.cpp hello1300v2.cpp
vocstartsoft:~/environment/rec2 $ ls
hello1300.cpp  hello1300v2.cpp
vocstartsoft:~/environment/rec2 $ 
```

Since we did not specify the path for the file we wanted to copy, nor for where we wanted it copied, we ended up with an identical copy of the file, both of them inside the folder `rec2`.

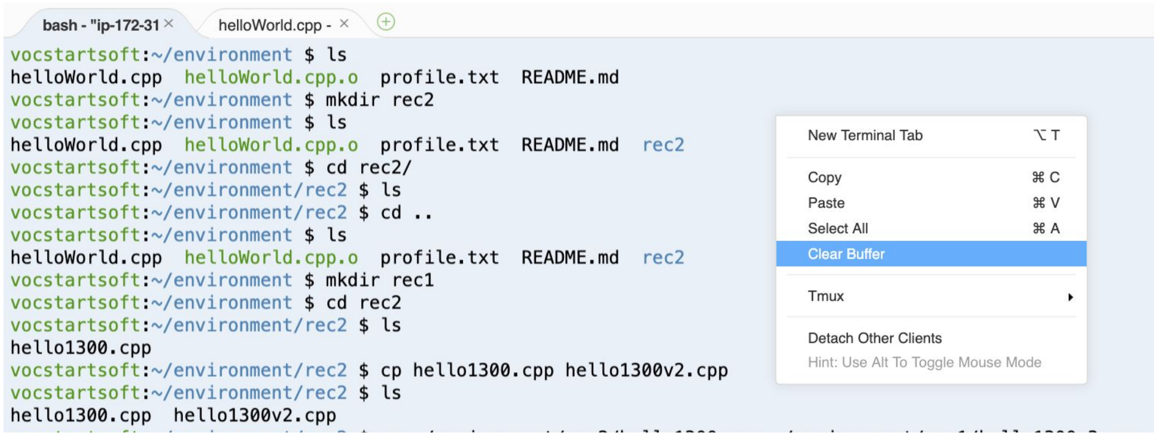
7. Let's do this again, including the paths:



```
vocstartsoft:~/environment/rec2 $ cp ~/environment/rec2/hello1300.cpp ~/environment/rec1/hello1300v3.cpp
```



8. If your terminal window has gotten full, you can always clear it. Right-click anywhere in it, and choose “Clear Buffer”



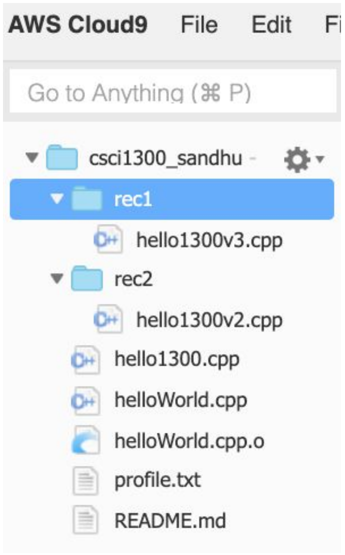
```
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md
vocstartsoft:~/environment $ mkdir rec2
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md  rec2
vocstartsoft:~/environment $ cd rec2/
vocstartsoft:~/environment/rec2 $ ls
vocstartsoft:~/environment/rec2 $ cd ..
vocstartsoft:~/environment $ ls
helloWorld.cpp  helloWorld.cpp.o  profile.txt  README.md  rec2
vocstartsoft:~/environment $ mkdir rec1
vocstartsoft:~/environment $ cd rec2
vocstartsoft:~/environment/rec2 $ ls
hello1300.cpp
vocstartsoft:~/environment/rec2 $ cp hello1300.cpp hello1300v2.cpp
vocstartsoft:~/environment/rec2 $ ls
hello1300.cpp  hello1300v2.cpp
```

`mv -- move files`

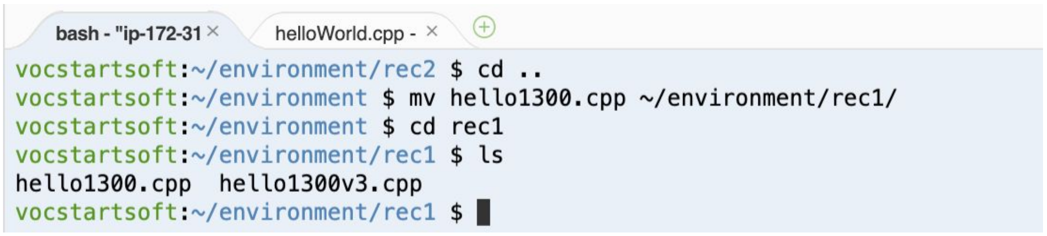
It’s possible to move files around using the command `mv`. Starting in directory `rec2`, type:

```
$ mv hello1300.cpp ..
```

Remember, two dots indicate the parent directory (one level up). Now if we list the files in `rec2`, there is only `hello1300v2.cpp` left. And we will move the file `hello1300.cpp` up in the root directory:



Let’s move one more file into the `rec1` directory:



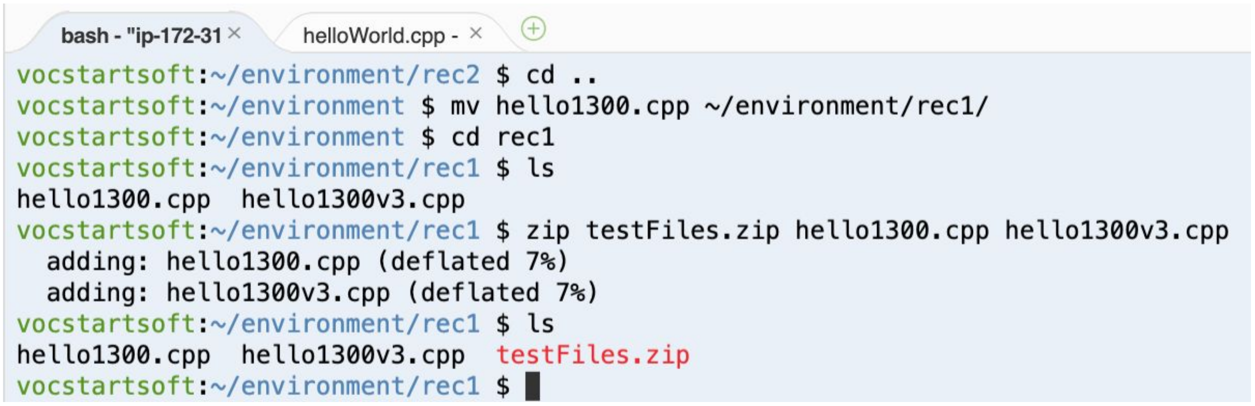
```
vocstartsoft:~/environment/rec2 $ cd ..
vocstartsoft:~/environment $ mv hello1300.cpp ~/environment/rec1/
vocstartsoft:~/environment $ cd rec1
vocstartsoft:~/environment/rec1 $ ls
hello1300.cpp  hello1300v3.cpp
vocstartsoft:~/environment/rec1 $ █
```

`zip` **(`unzip`)** `- package and compress (archive) files`

An important thing to know for this class is how to zip your solution files into one assignment submission file. To do this, there's a convenient command called zip. Let's zip up the two files in the directory `rec1` into a single zip file.

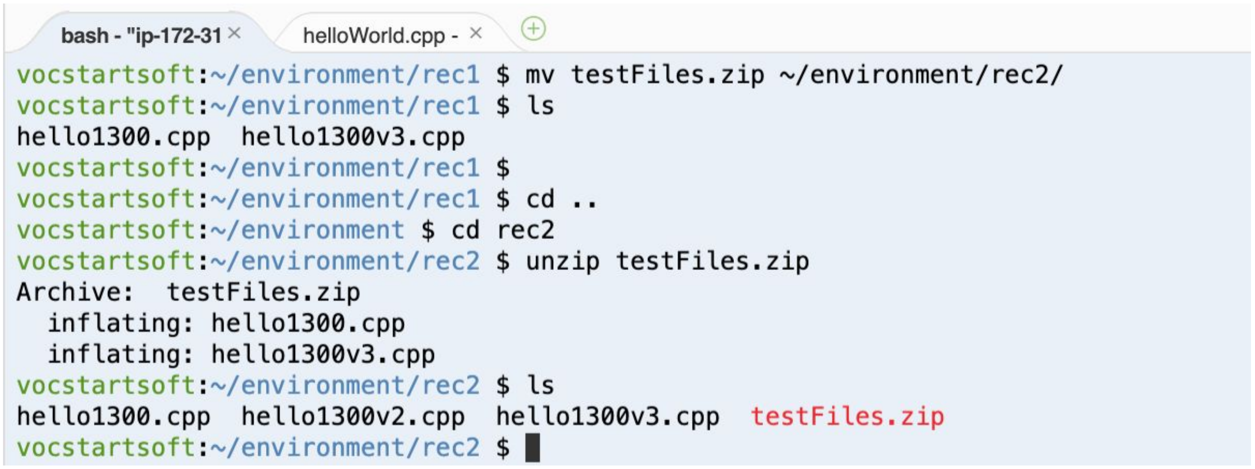
```
$ zip test_files.zip  hello1300.cpp hello1300v3.cpp
```

The first argument is the name of the zip file we want to produce. The files listed next are the files that go into this zip file. If you list your directory now, you should see three files. Notice that the original files are still present--zip and unzip don't destroy any files.



```
vocstartsoft:~/environment/rec2 $ cd ..
vocstartsoft:~/environment $ mv hello1300.cpp ~/environment/rec1/
vocstartsoft:~/environment $ cd rec1
vocstartsoft:~/environment/rec1 $ ls
hello1300.cpp  hello1300v3.cpp
vocstartsoft:~/environment/rec1 $ zip testFiles.zip hello1300.cpp hello1300v3.cpp
  adding: hello1300.cpp (deflated 7%)
  adding: hello1300v3.cpp (deflated 7%)
vocstartsoft:~/environment/rec1 $ ls
hello1300.cpp  hello1300v3.cpp  testFiles.zip
vocstartsoft:~/environment/rec1 $ 
```

Let's now move the .zip file into the `rec2` directory, and try to unzip the files:



```
vocstartsoft:~/environment/rec1 $ mv testFiles.zip ~/environment/rec2/
vocstartsoft:~/environment/rec1 $ ls
hello1300.cpp  hello1300v3.cpp
vocstartsoft:~/environment/rec1 $
vocstartsoft:~/environment/rec1 $ cd ..
vocstartsoft:~/environment $ cd rec2
vocstartsoft:~/environment/rec2 $ unzip testFiles.zip
Archive:  testFiles.zip
  inflating: hello1300.cpp
  inflating: hello1300v3.cpp
vocstartsoft:~/environment/rec2 $ ls
hello1300.cpp  hello1300v2.cpp  hello1300v3.cpp  testFiles.zip
vocstartsoft:~/environment/rec2 $ 
```

You should see the three files in the directory. Note that the original zip file didn't get destroyed.

Linux commands we've learned:

Here are the five useful Linux commands we just practiced:

- **ls**: List directory contents. Use this command to see the list of files and folders in the current directory.
- **mkdir**: Make directories. Create a new directory (folder).

- **cd**: Change directory. Use this command to change the current directory (move to another folder)
- **cp**: Copy a file
- **mv**: Move a file
- **zip**: Create a zip file

Fun tips and tricks

- **Tab Complete:** if you're typing something in the command line that's very long, but unique, you can hit tab when you're partially through and it will try to fill in the rest (kind of like autocomplete). If it doesn't, and you tap tab twice, it tells you everything it has as options.
- **Command history browsing:** if you have typed a command (e.g. `gedit myFile.txt`) and want to repeat it, just press the up arrow. It will bring up your last executed command. Pressing up again will go to the one before. Pressing down will go forward in time through the list.
- ASCII is a character encoding standard. It includes numerical characters: 0-9, letters A-Z and a-z, punctuation, and blank spaces. These character codes represent text in computers. For example, the ASCII representation of 'D' is 01000100, and the ASCII representation of 'd' is 01100100. Notice the difference? 'D' is different from 'd', therefore, it is important for you to always make sure that you have proper capitalization and spelling when you are typing from the command line.

Want to learn more about Linux commands?
http://community.linuxmint.com/tutorial/view/244 has a list of categorized Linux commands.

Codecademy also has a course covering the command line (https://www.codecademy.com/learn/learn-the-command-line).

2.7 **Setup for Visual Studio Code**:

You also need to set up a development environment locally to make sure that you can write code without an internet. Also, in case Cloud 9 goes down for maintenance, you still need to work and complete assignments.

You can find the VS code set up guide on Moodle. Follow the instructions and set up it locally.

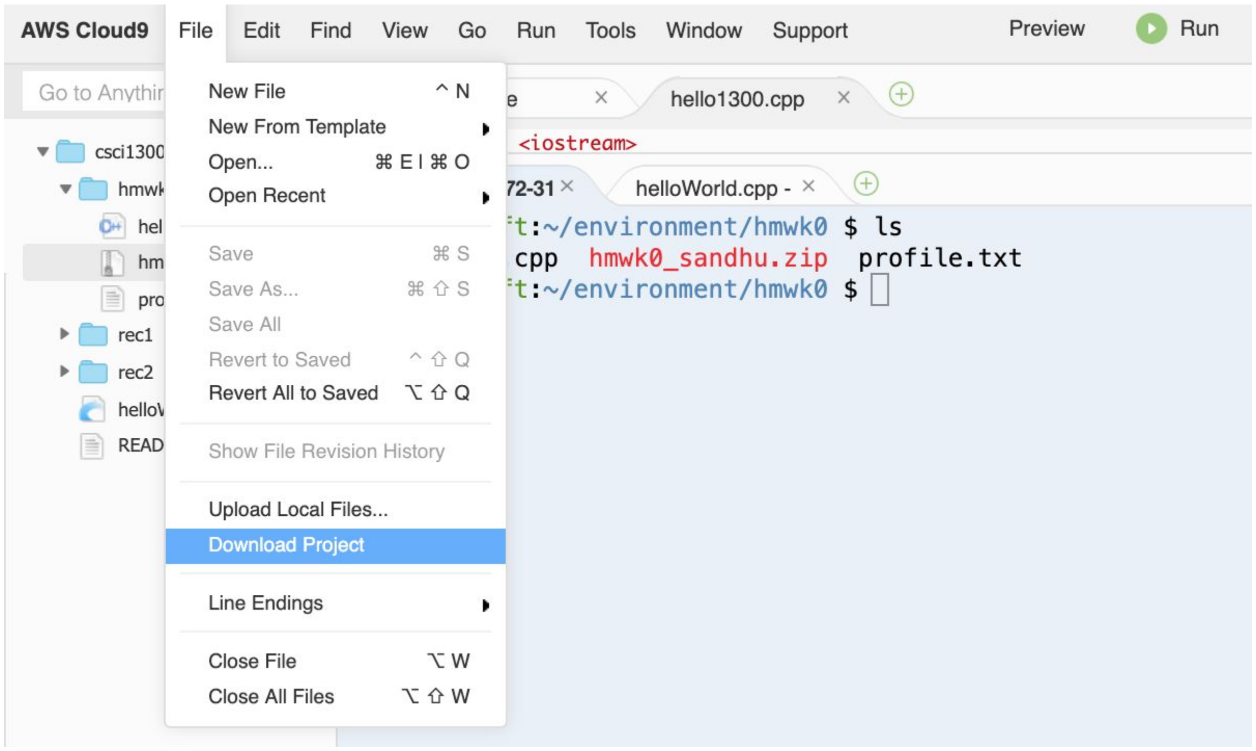
Task3: Write a helloWorld in VS code as you did in section 2.4. Your program should print "`Hello CSCI1300 from VS code`". Be sure that your code compiles and runs on both your machine and cloud 9. Your .cpp file should be named "`helloWorldVScode.cpp`"

2.8 Back up often:

It’s a good idea to create a backup of your workspace. Backing up a file only takes a few seconds. You will hate yourself if you lose hours of work before the deadline. We recommend that you backup your work once every thirty minutes.

Cloud 9

In case, Cloud 9 goes down for maintenance unexpectedly be sure to have a backup. You can simply download an entire environment.



```
t:~/environment/hmwk0 $ ls
 cpp   hmwk0_sandhu.zip   profile.txt
t:~/environment/hmwk0 $ 
```

Your computer

It’s also important to backup your computer in case your computer gets broken or your dog bites your laptop. You can use:

- Dropbox
- Google Drive
- Github (be sure to use Private repository. You can get free private repository via GitHub Student Developer Pack)

2.7 **Submit your work to Moodle**:

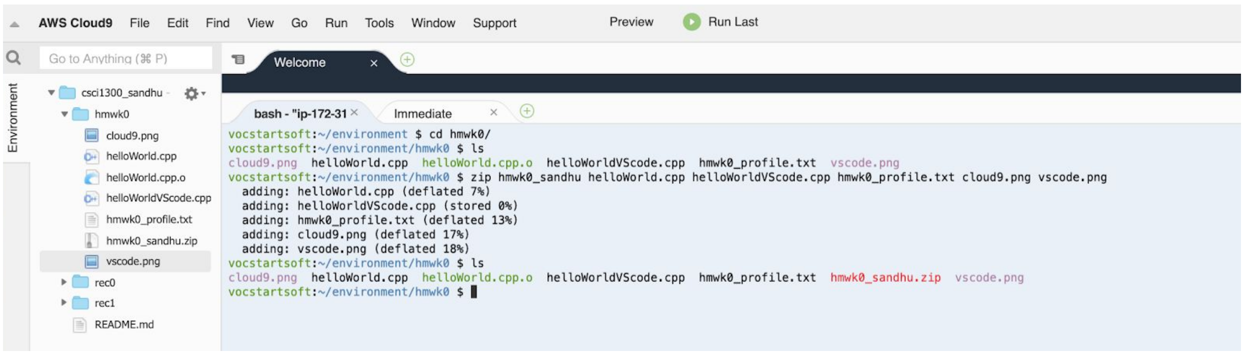
Screenshots

We would like to make sure that you set up both Cloud 9 and VS code on your machine. Take screenshots of windows for Cloud 9 and VS code, and name them **cloud9.png** (or **cloud9.jpg**) and **vsCode.png** (or **vsCode.jpg**).

Zip your submission file

Create a new directory called, **hmwk0**. In the directory, put your **profile.txt** and **helloWorld.cpp, helloWorldVScode.cpp, cloud9.png, and vsCode.png**. Then, create a zip file with them and name it **hmwk0_yourLastName.zip**. If your last name is **Bezos**, then the name of the zip file should be **hmwk0_Bezos.zip**

On your cloud 9 environments, you should have **hmwk0** folder and the files you're going on to submit:

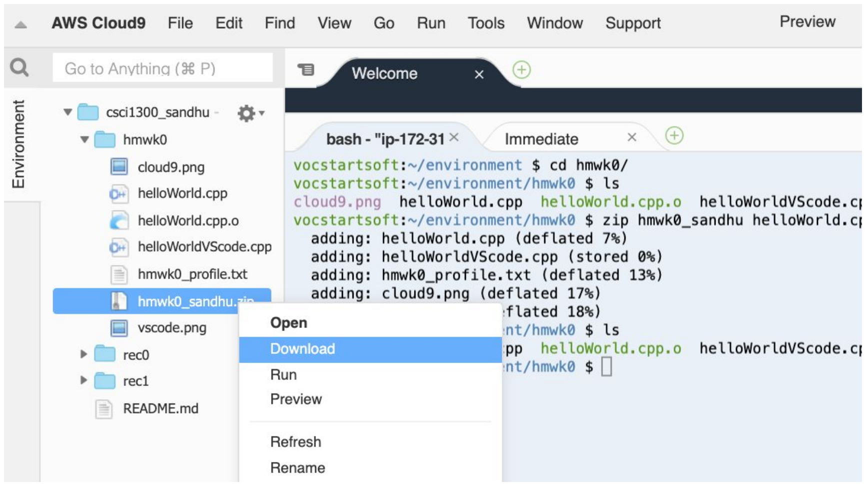


```
vocstartsoft:~/environment $ cd hmwk0/
vocstartsoft:~/environment/hmwk0 $ ls
cloud9.png  helloWorld.cpp  helloWorld.cpp.o  helloWorldVScode.cpp  hmwk0_profile.txt  vscode.png
vocstartsoft:~/environment/hmwk0 $ zip hmwk0_sandhu helloWorld.cpp helloWorldVScode.cpp hmwk0_profile.txt cloud9.png vscode.png
  adding: helloWorld.cpp (deflated 7%)
  adding: helloWorldVScode.cpp (stored 0%)
  adding: hmwk0_profile.txt (deflated 13%)
  adding: cloud9.png (deflated 17%)
  adding: vscode.png (deflated 18%)
vocstartsoft:~/environment/hmwk0 $ ls
cloud9.png  helloWorld.cpp  helloWorld.cpp.o  helloWorldVScode.cpp  hmwk0_profile.txt  hmwk0_sandhu.zip  vscode.png
vocstartsoft:~/environment/hmwk0 $
```

(You might have other directories or files(such as `.cpp.o`). That's good! Don't forget to keep your workspace organized!)

Download your zip file

Once you create a zip file on Cloud 9, you can download it by choosing a file and right-click to download it

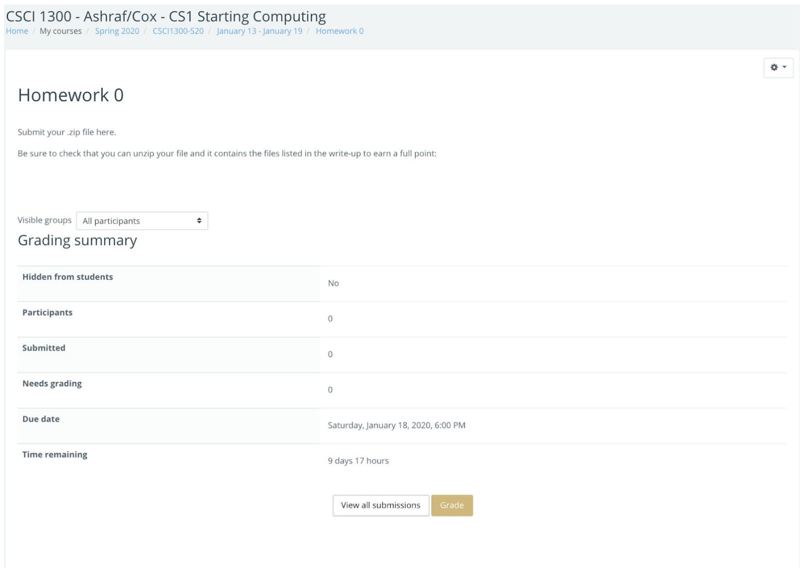


```
vocstartsoft:~/environment $ cd hmwk0/
vocstartsoft:~/environment/hmwk0 $ ls
cloud9.png  helloWorld.cpp  helloWorld.cpp.o  helloWorldVScode.cp
vocstartsoft:~/environment/hmwk0 $ zip hmwk0_sandhu helloWorld.cp
  adding: helloWorld.cpp (deflated 7%)
  adding: helloWorldVScode.cpp (stored 0%)
  adding: hmwk0_profile.txt (deflated 13%)
  adding: cloud9.png (deflated 17%)
                        flated 18%)
                        nt/hmwk0 $ ls
                        pp  helloWorld.cpp.o  helloWorldVScode.cp
                        nt/hmwk0 $
```

.

Submit it to Moodle

Once you download the zip file, go to hmwk0 on Moodle, click “add submission”, and upload it on the Moodle submission.



CSCI 1300 - Ashraf/Cox - CS1 Starting Computing

Home / My courses / Spring 2020 / CSCI1300-S20 / January 13 - January 19 / Homework 0

Homework 0

Submit your .zip file here.

Be sure to check that you can unzip your file and it contains the files listed in the write-up to earn a full point:

Visible groups: All participants

Grading summary

Hidden from students	No
Participants	0
Submitted	0
Needs grading	0
Due date	Saturday, January 18, 2020, 6:00 PM
Time remaining	9 days 17 hours

Task4: Download your **hmwk0_yourLastName.zip** and submit it to hmwk0 on Moodle.

3. Homework 0 points summary

Criteria	**Pts**
Zip file submission	50
Recitation attendance (in week 1, Jan 13)*	-20
Total	50

* if your attendance is not recorded, you will lose points. Make sure your attendance is recorded on Moodle.