CSCI 1300 CS1:Starting Computing
Recitation 1
Instructor: Fleming
Due: Friday, September 7th at 11:55 pm

Objectives:

1. Become familiar with your Integrated Development Environment (Cloud9).
2. Write your first C++ program.
3. Become familiar with testing and submitting your solution for auto-grading on Moodle.

Readings:

Book: "Brief C++ Late Objects, Third Edition, Cay S. Horstmann, Wiley. 2017. ISBN: 978-1118674260"

1. Read chapters 1.5 and 1.6 before Recitation 1.
2. Read chapters 2.1, 2.2, 2.3, and 2.4 before the lectures for the week of 9/3.

Cloud 9 IDE, continued:

In Recitation 0, you created an account with Cloud9 and spun off your Private Workspace. One of the cool things about Cloud 9 is that every time you open your Private Workspace, it should look exactly like you left it when you closed your browser.

Open your Private Workspace and you should have the default files, two folders (rec0 and rec1), a few tabs at the bottom with the last commands we ran for unzipping files.

Running applications from the command line

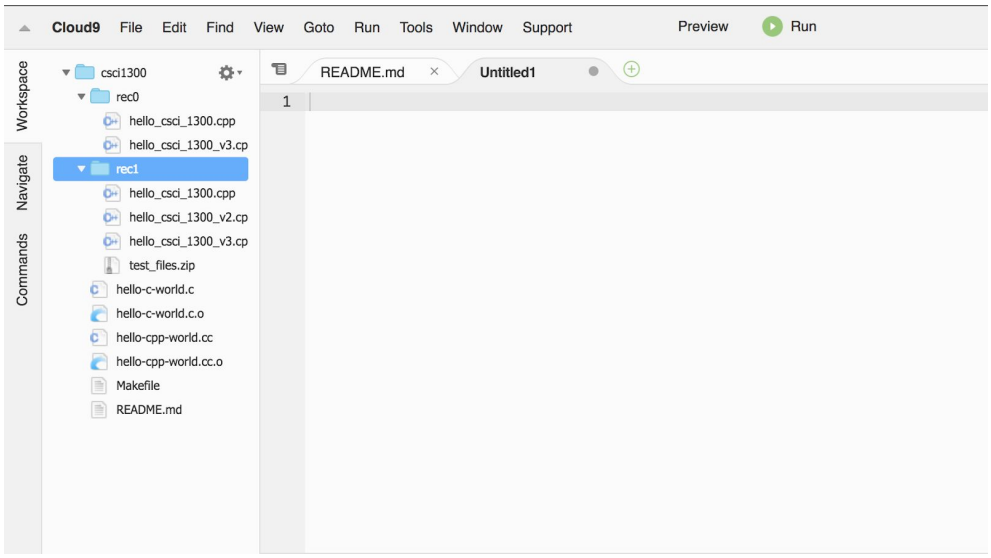
Many of the applications in Cloud9 can be run from the terminal. Running programs in the terminal is also called running them *from the command line*. In this recitation, you will run your first C++ program, run it from the command line, then copy the solution and submit it on Moodle for autograding.

Hello, World

The first program that we usually write in any language we're learning is Hello, World. A Hello, World program just prints "Hello, World!" to the screen.

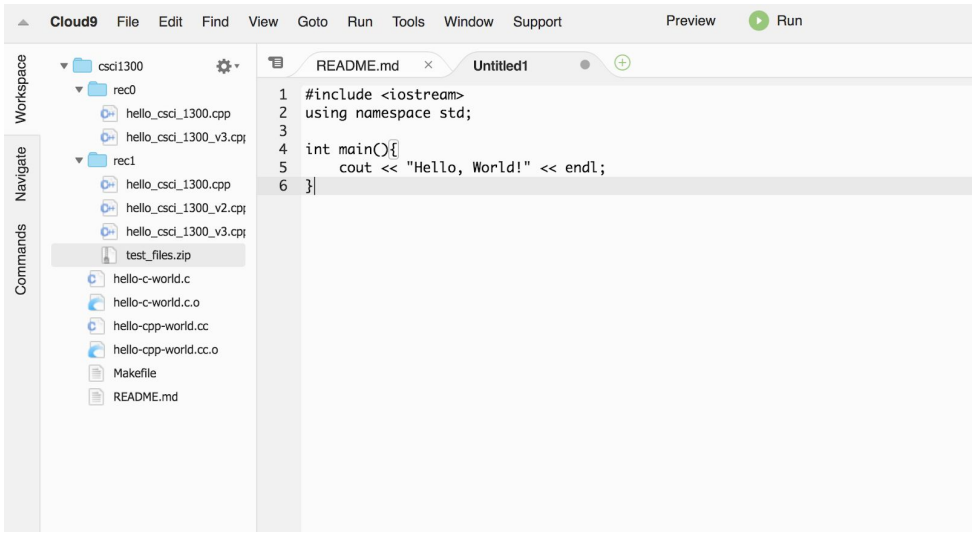
Step 1: Open an Empty File

In Cloud9, select `File -> New -> New File`. A new, blank file called Untitled1 will be opened.



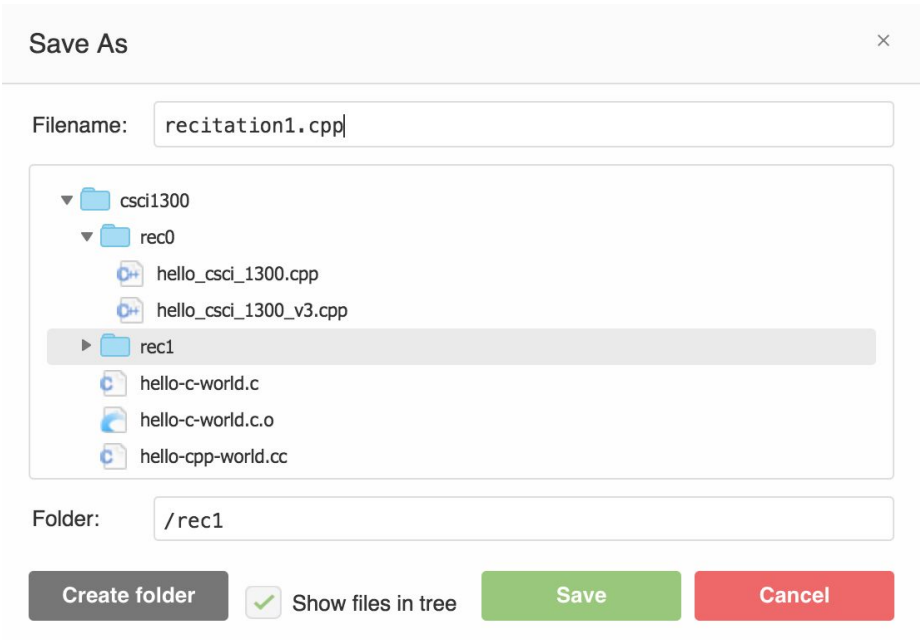
Step 2: Your First Code!

Starting on line 1 in Untitled1, type the following code:

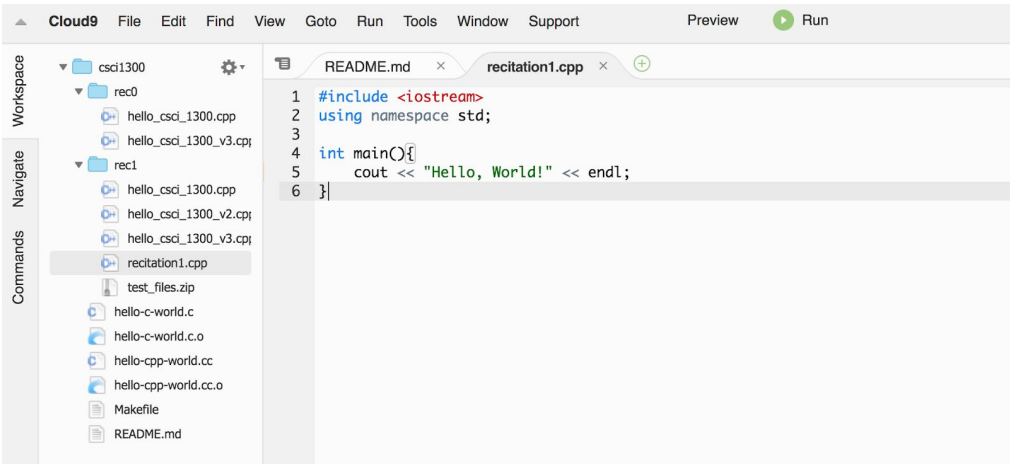


Step 3: Saving Your File

Save the file: go to `File -> Save As...` A dialog box will open. Name it **recitation1.cpp** and save it in the rec1 folder.



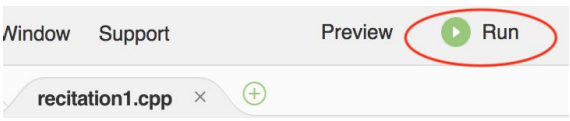
The .cpp extension on the filename tells Cloud9 that the file should be read in the C++ programming language. Once you save it, the lines in the file should be color-coded to reflect what they do in the program. This is called *syntax highlighting*.



Important: You should save your work frequently in Cloud9 to avoid losing your work in the event of the program crashing.

Step 4: Running Your Code

To run the program, click on the icon with the green arrow next to the word Run. If it works, you should see new terminal tab window open at the bottom.



The title of the tab shows the file being run (rec1/recitation1.cpp), and inside the window you should see “Running ....” (again the name and full path of the file), and underneath it, the output of our program: Hello, world!

Step 5: Running Your Code from Command Line

Move to the “bash” tab (the first tab in the bottom panel). Right-click again and Clear the Buffer. Make sure you are inside the *rec1* directory. Type:

```
$ g++ recitation1.cpp -g -std=c++11
```

the –g option turns on debugging, which we will use later in the semester, so we should get used to it.

the `-std=c++11` option makes sure that the c++ version used to run the program is c++ 11. If you don’t give this option then default version(which is usually C++98) is used.

This creates an executable called "a.out" (see figure below). You can run it by typing

```
$ ./a.out
```

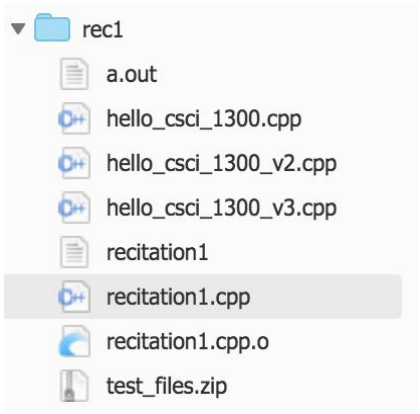
Since no executable name was specified to g++, a.out is chosen by default. You can alternatively use the "-o" option to change the name :

```
$ g++ recitation1.cpp -g -std=c++11 -o recitation1
```

creates an executable called "recitation1" (see figure below). You can run it by typing

```
$ ./recitation1
```

Notice the output in the same: Hello, world!, followed by the return of the prompt, for new commands.




```
bash - "ubuntu@
tellyumada:~/workspace $ cd rec1/
tellyumada:~/workspace/rec1 $ g++ recitation1.cpp -g -std=c++11
tellyumada:~/workspace/rec1 $ ./a.out
Hello, World!
tellyumada:~/workspace/rec1 $ g++ recitation1.cpp -g -std=c++11 -o recitation1
tellyumada:~/workspace/rec1 $ ./recitation1
Hello, World!
tellyumada:~/workspace/rec1 $
```

Step 6: Update Your Code

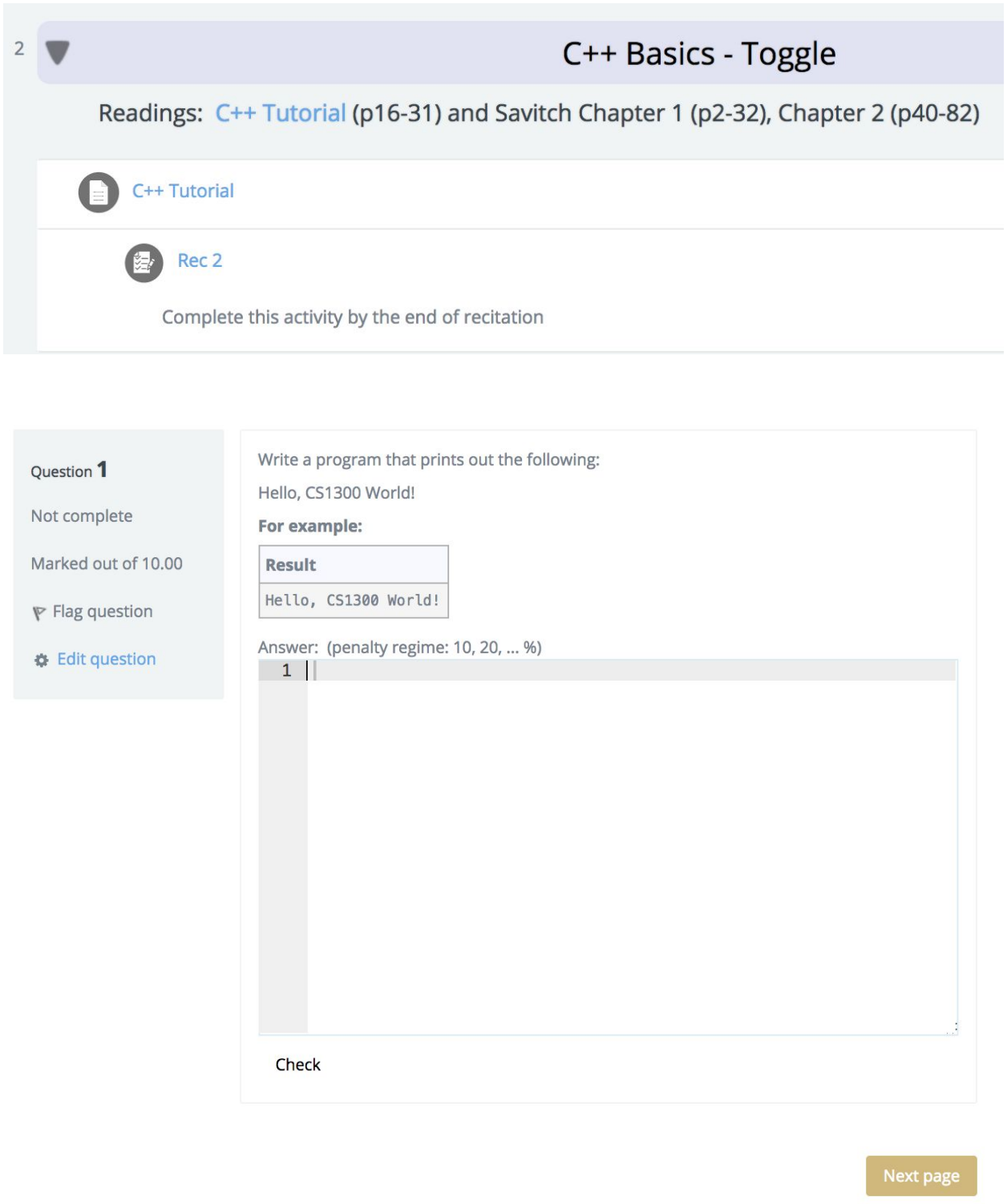
Now, change the `cout` statement so that your program prints out the following:

Hello, CS1300 World!

```
bash - "ubuntu@
tellyumada:~/workspace/rec1 $ g++ recitation1.cpp -g -std=c++11 -o recitation1
tellyumada:~/workspace/rec1 $ ./recitation1
Hello, CS1300 World!
tellyumada:~/workspace/rec1 $
```

Step 7: Submitting Your Code on Moodle

Throughout the semester we will be using the Moodle autograder. Take a look at the first question from the *Rec1* activity on Moodle:



You can see:

1. The description of the problem:
   Write a program that prints out the following:
   Hello, CS1300 World!

2. An example of the program's output:

For example:

Result
Hello, CS1300 World!

3. The Answer box – this is where you will paste your code
4. The "Check" button – you can click this button as many time as you wish to check if the solution/code you entered in the Answer box is correct
5. On the left side you can see the problem is worth 10 points, and
6. At the bottom right there is a button "Next Page" which will lead you to the next problem

What you need to do:

1. Copy your code from Cloud9 from the updated file *recitation1.cpp*
2. Paste it into the Answer box:

Answer: (penalty regime: 10, 20, ... %)

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hello, CS1300 World!";
7  }
```

3. Press the Check button. If you code is correct, and the output from your solution matches the expected output, you should see green check marks indicating you have passed the test:

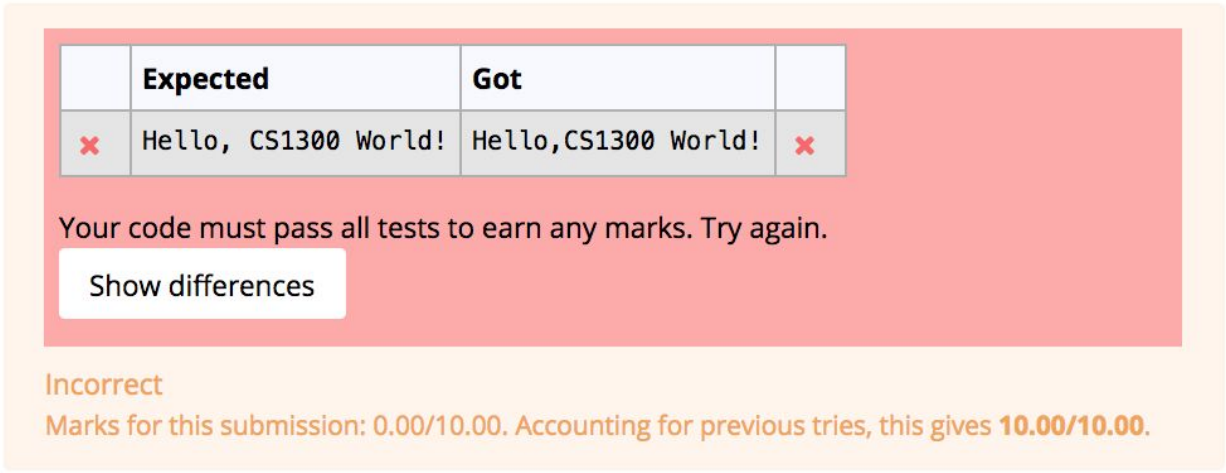
	Expected	Got	
✔	Hello, CS1300 World!	Hello, CS1300 World!	✔

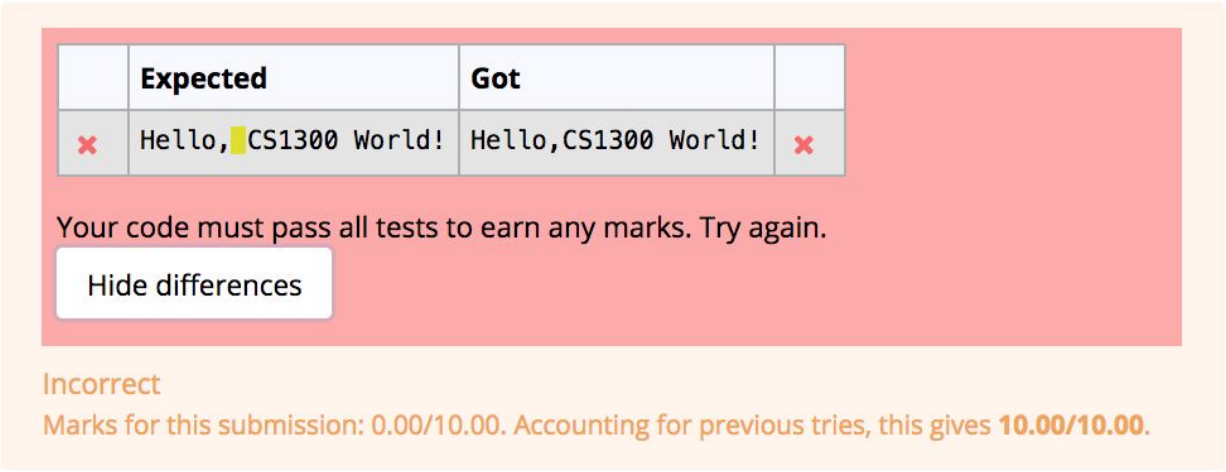
Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

4. In the event you have a typo in the output message, the test will show the Expected output and the output from your solution (in the Got column). Look at the differences; update your solution to match the Expected output. The button Show Differences should help identify what is missing (or extra). In the example below, the solution forgot to include a "space" after the comma:





Note: notice that since the first attempt was correct, the problem is still scored at 10 points, "Accounting for previous tries, this gives 10.00/10.00"

Good Practice: backup your work!

It is recommended you use a backup system for your work. Get a Dropbox account or use Google Drive. Dropbox accounts are limited, while a CU Google Drive allows up to 1TB (that should be enough for all your work, for all the courses you will take at CU).

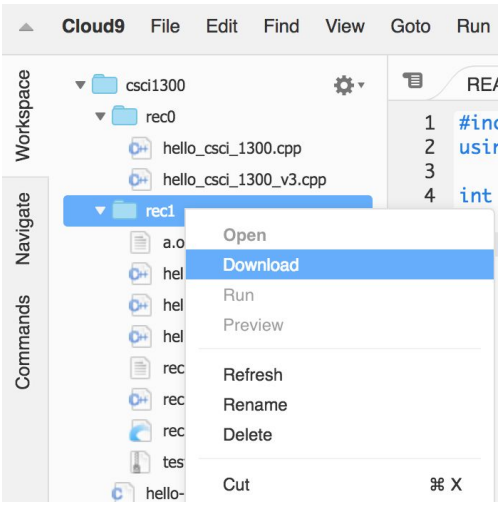
In order to backup our work, we first need to bring it out from Cloud9. It's very simple, and you have multiple options:

1. Download just one file (the solution file you just submitted).

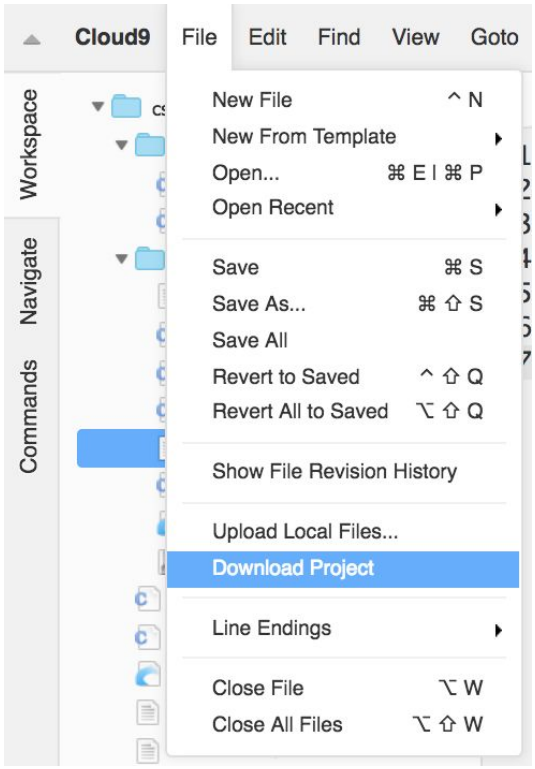
   Right click on the file name in the left panel (the file tree) and choose the option Download. Only one file will be downloaded, on your computer, at your default download location (your Downloads or My Downloads folder)

2. Download the entire *rec1* folder.

   Right click on the folder's name and choose the option Download. An archive .zip file will be downloaded on your computer, with all the files from the *rec1* folder.



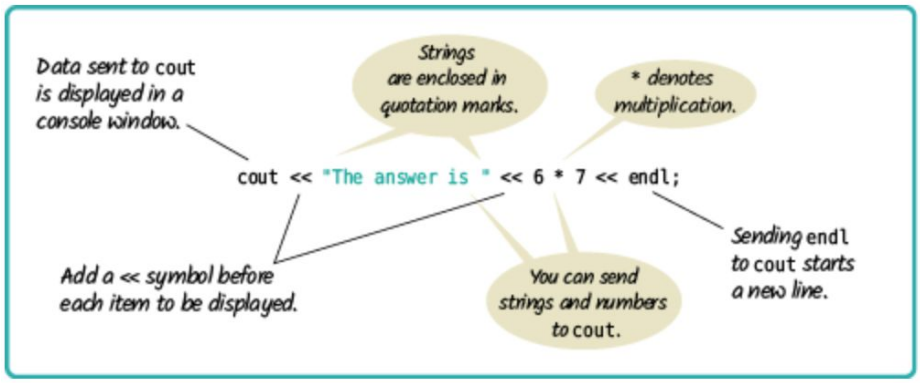
3. At any time, you can download the entire Cloud9 workspace. This means all the files you have created in Cloud9 since you first logged in. Go to File -> Download Project . This will download an archive .zip file, containing a folder named "Workspace", which includes all the subfolders you created, with all the files.



Basic input/output

cout:
The cout operator inserts the data into the console and it is written as <<. Multiple insertion operations (<<) may be chained in a single statement. For example,

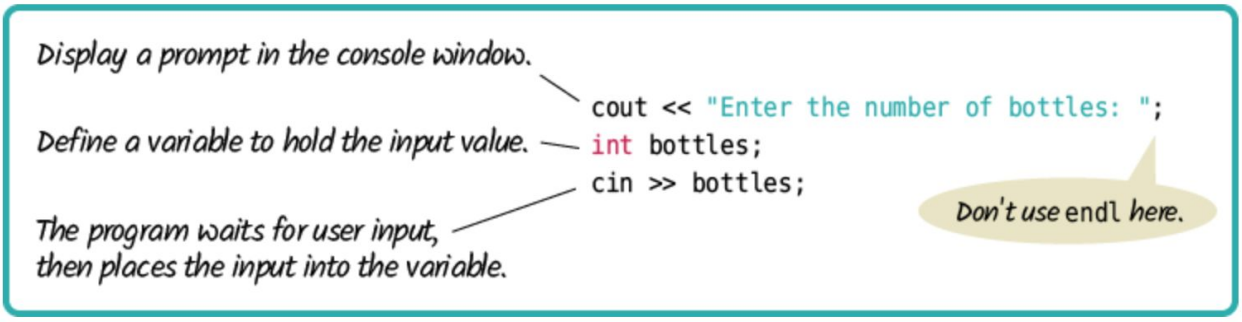
cout << "This " << "is a " << "single C++ statement"<<endl;



cin:
cin is used together with the extraction operator, which is written as >>. This operator is then followed by the variable where the extracted data is stored. When the program executes the input statement, it waits for the user to provide input. The user also needs to press the Enter key so that the program accepts the input. For example,

int number;
cin>>number;

When a program asks for user input, it should first print a message that tells the user which input is expected. Such a message is called a prompt.



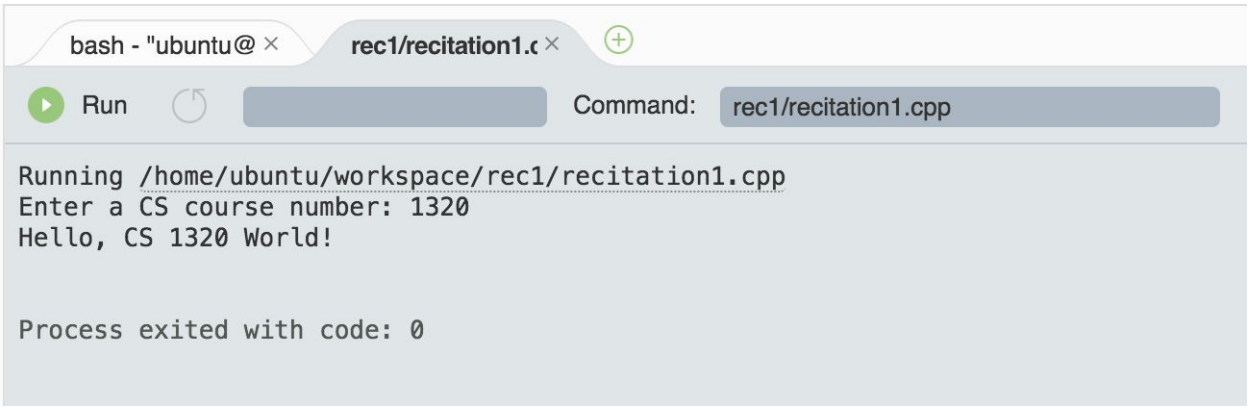
Other Moodle Programming Questions

Problem 2 is the same as Problem 1, except that you will be utilizing a variable of type integer. Write a program that prints out the same output as Problem 1, but include an integer variable to store the value of course number as given by the user as input and print that course number.

You will first need to prompt the user to enter a CS course number with a statement like "Enter a CS course number". Once the user enters the course number, print the required output.

For example:
If the user enters the input "*1320*", the output of the program should be,

Hello, CS 1320 World!



Remember: you can press Check as many times as you want. It will show you what the Expected Output is, and also what is the output produced by your code. Compare the two outputs and make adjustments to your code. Use the Show Differences button to help you figure out what's missing, or what you might have extra. Good luck!

Once you are done with the activity raise your hand and show your work to the TA.