CSCI 1300 - Intro to Computer Programming
Instructor: Fleming
Recitation 2

This assignment is due **Friday, September 14th at 11:55 pm**

- ***All components* (Cloud9 workspace and moodle quiz attempts) must be completed and submitted by Friday, September 14th at 11:55 pm for your solution to receive points.**
- **Recitation attendance is required to receive credit.**

Objectives:

- Understand basic programming concepts of: variables, values, assignment, operations, operands, and use them in simple C++ commands
- Understand and practice using `cout` to display text and variables (their values)
- Understand and practice using `cin` to take input from the user
- Write and test short C++ functions
    - write functions with a single input parameter
    - write functions that do not have return values

Please follow the same submission guidelines outlined in Homework 2 description regarding Style, Comments and Test Cases. Here's a review below on what you need to submit for Recitation 2.

Develop in Cloud9: For this recitation assignment, write and test your solution using Cloud9.
Submission: Both steps must be fully completed by the submission deadline for your homework to receive points. Partial submissions will not be graded.

1. ***Share your Cloud 9 workspace with your TA:*** Your recitation TA will review your code by going to your Cloud9 workspace. *TAs will check the last version that was saved before the submission deadline*.
    - Create a directory called **Rec2** and place all your file(s) for this assignment in this directory.
    - Share your workspace by clicking Share in the upper right hand corner and inviting your TA using their CU email address (see the syllabus).
    - Make sure to *save* the final version of your code (File > Save). Verify that this version displays correctly by going to File > File Version History.
    - The file(s) should have all of your functions, test cases for the functions in main function(s), and adhere to the style guide. Please read the **Test Cases** and **Style and Comments** sections for more details.
2. ***Submit to the Moodle Autograder:*** Head over to Moodle to the link **Rec 2 CodeRunner**. You will find one programming quiz question for each problem in the assignment. Submit your solution for the first problem and press the Check button. You will see a report on how your solution passed the tests, and the resulting score for the first problem. You can modify

your code and re-submit (press *Check* again) as many times as you need to, up until the assignment due date. Continue with the rest of the problems.

Problems Set:
Write a **function** for each of the following problems. You should first write your solution in Cloud9. Then copy and paste your function into the Moodle quiz questions. The code you write on Cloud9 and the code you copy and paste into Moodle must include an algorithm and comments.

Problem 1

Alter the provided main to print both the volume and surface area of a sphere with given radius. For a radius of 5, the output of the program should look like this:

```
Enter a radius:
5
volume: 523.599
surface area: 314.159
```

Remember that the surface area of a sphere with radius **r** is $\mathbf{4\pi r^2}$.

Problem 2

Write a function called **sphereVolume** that determines the volume of a sphere with given radius and prints the result to the screen.

- Your function should have **one** input value:
    - a floating point parameter representing the radius
- Your function should not return anything.
- Your function should **print** the calculated volume.
    - The output format should resemble that of the previous problem. For a radius of 5, the function should print: `volume: 523.599`
- Your function *MUST* be named **sphereVolume**.

Problem 3

Write a function called **sphereSurfaceArea** that determines the surface area of a sphere with given radius and prints the result to the screen.

- Your function should have **one** input value:
    - a floating point parameter representing the radius
- Your function should not return anything.
- Your function should **print** the calculated surface area.

- ○ The output format should resemble that of the previous problem. For a radius of 5, the function should print: `surface area: 314.159`
- Your function *MUST* be named **sphereSurfaceArea.**

Problem 4

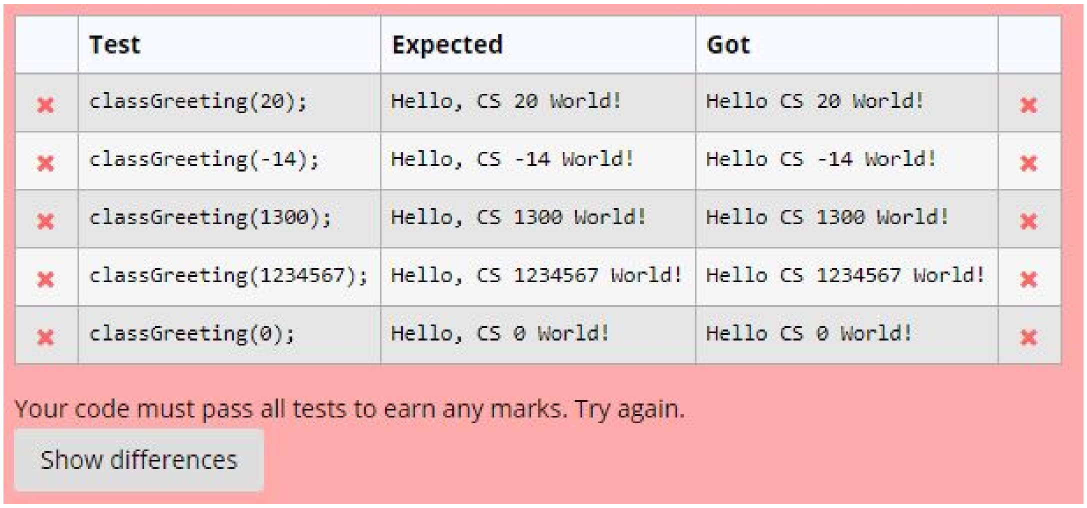
Convert the second problem from last week's recitation to a function. Specifically, write a function called **classGreeting** that takes a single integer argument and prints a greeting to the screen. If 1300 is the given value, the function should print:

```
Hello, CS 1300 World!
```

Submitting Your Code to the Autograder on Moodle

You must name the functions as indicated in each problem description. **Importantly**, *the cout formats provided for each problem are not suggestions – they **MUST** be followed precisely, word-for-word and including all punctuation marks*, otherwise the autograder will not recognize your results and you will not receive credit.

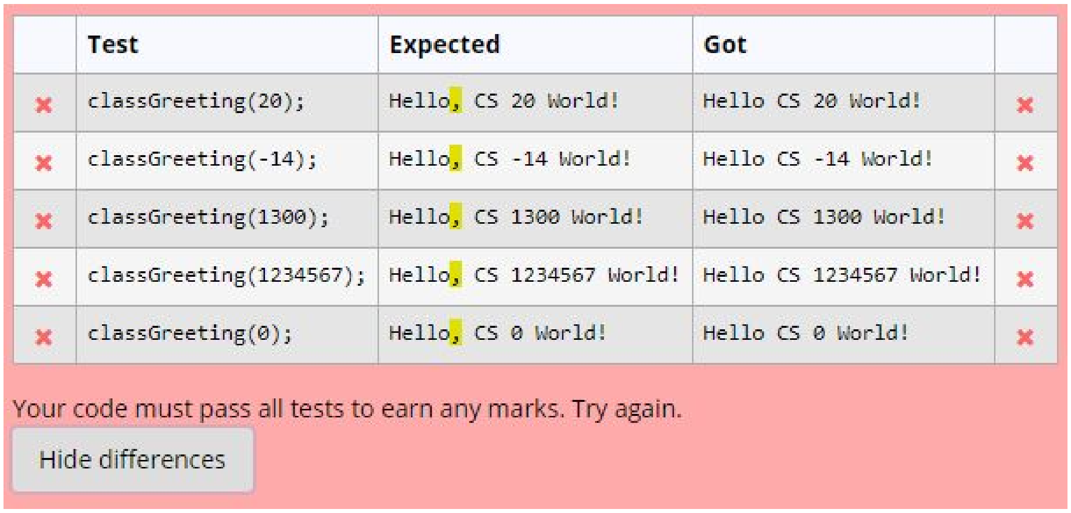
If there are errors in your solution to a particular problem, a button labeled "Show differences" will appear below the table of tests after you hit "check". This can be a very useful tool in helping you find small typos, especially in cout statements.



	Test	Expected	Got	
✗	classGreeting(20);	Hello, CS 20 World!	Hello CS 20 World!	✗
✗	classGreeting(-14);	Hello, CS -14 World!	Hello CS -14 World!	✗
✗	classGreeting(1300);	Hello, CS 1300 World!	Hello CS 1300 World!	✗
✗	classGreeting(1234567);	Hello, CS 1234567 World!	Hello CS 1234567 World!	✗
✗	classGreeting(0);	Hello, CS 0 World!	Hello CS 0 World!	✗

Your code must pass all tests to earn any marks. Try again.

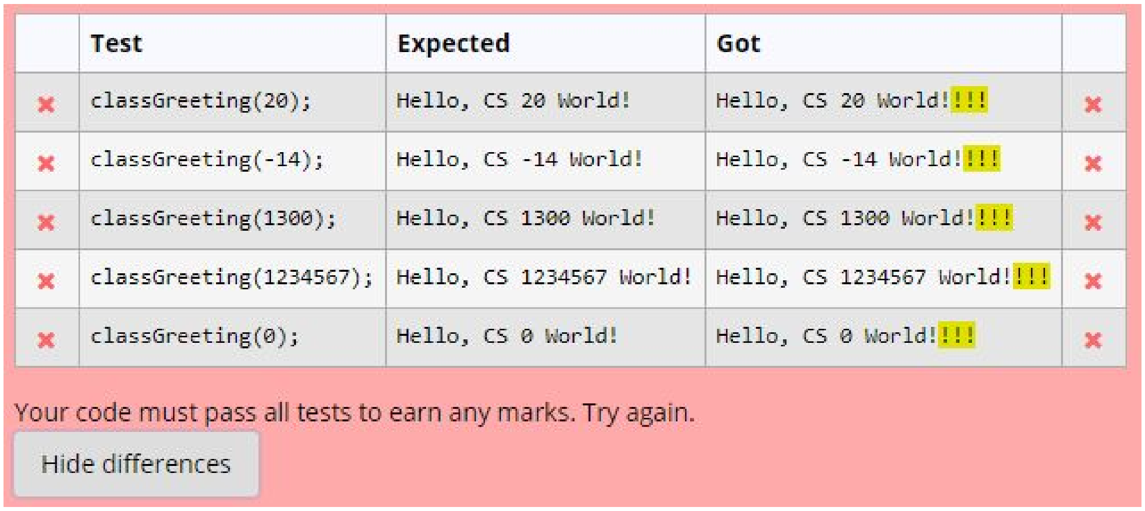
For example, below we hit "check" for a solution to problem 1 on this recitation and have failed all the test cases despite getting the correct values. Hitting "Show differences", we can see that a comma (,) is missing. When characters are in the expected output but not in your output they are highlighted in the "Expected" column.



	Test	Expected	Got	
✖	`classGreeting(20);`	`Hello, CS 20 World!`	`Hello CS 20 World!`	✖
✖	`classGreeting(-14);`	`Hello, CS -14 World!`	`Hello CS -14 World!`	✖
✖	`classGreeting(1300);`	`Hello, CS 1300 World!`	`Hello CS 1300 World!`	✖
✖	`classGreeting(1234567);`	`Hello, CS 1234567 World!`	`Hello CS 1234567 World!`	✖
✖	`classGreeting(0);`	`Hello, CS 0 World!`	`Hello CS 0 World!`	✖

Your code must pass all tests to earn any marks. Try again.

On the other hand, when we include extra, unexpected characters in output they are highlighted in the “Got” column. Below we added additional exclamation points (!) to the output.



	Test	Expected	Got	
✖	`classGreeting(20);`	`Hello, CS 20 World!`	`Hello, CS 20 World!!!!`	✖
✖	`classGreeting(-14);`	`Hello, CS -14 World!`	`Hello, CS -14 World!!!!`	✖
✖	`classGreeting(1300);`	`Hello, CS 1300 World!`	`Hello, CS 1300 World!!!!`	✖
✖	`classGreeting(1234567);`	`Hello, CS 1234567 World!`	`Hello, CS 1234567 World!!!!`	✖
✖	`classGreeting(0);`	`Hello, CS 0 World!`	`Hello, CS 0 World!!!!`	✖

Your code must pass all tests to earn any marks. Try again.