CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2

Due Sunday, September 16th, by 6 pm
+5% bonus if submitted by Friday September 14th 11:55 pm,
+2% bonus if submitted by Saturday September 15th 11:55 pm

This assignment is due **September 16th, by 6 pm**

- ***All components*** **(Cloud9 workspace, moodle quiz attempts, and zip file) must be completed and submitted by Sunday, September 16th, 6:00 pm for your homework to receive points.**
- Complete submissions (Cloud9 workspace, moodle quiz attempts, and zip file) before **Friday September 14th 11:55 pm** will receive a 5% bonus, and complete submissions before **Saturday September 15th 11:55 pm** will receive a 2% bonus.

Objectives:

- Algorithm Design: be able to write a step-by-step pseudocode algorithm to a given problem
- 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)
- Understanding difference between `cout` and `return` statement
- Writing and testing C++ short functions
    - Understand problem description
    - Design your function:
        - come up with a step by step algorithm,
        - convert the algorithm to pseudocode
        - imagine many possible scenarios and corresponding sample runs or outputs
    - Convert the pseudocode into a program written in the C++ programming language
    - Test it in the Cloud9 IDE and submit it for grading on Moodle

---

Develop in Cloud9: For this assignment, write and test your solution using Cloud9.

Submission: All three 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*.

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2

- Create a directory called **Hmwk2** 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 Cloud9 username.

TA Name	Cloud9 Username
Gabriel Andrade	gpandrade
Paramjot Singh	param17
Tetsumichi(Telly) Umada	tetsumichiumada
Richard Tillquist	Carter
Ashwin Sankaralingam	ashwinsankaralingam
Shudong Hao	shudong
Andrew Altomare	andrewaltomare
Isabella Huang	isabellahuang4

TA Name	Cloud9 Username
Varsha Koushik	varshak
John Klingner	john.klingner@colorado.edu
Sebastian Laudenschlager	slaudens
Dixit Patel	dixitpatel
Harshini Muthukrishnan	harshini95
Lucas Hayne	luchayne
Chu Sheng Ku	chusheng
Supriya M. Naidu	supriyanaidu

- 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 **Hmwk 2**. 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.

3. ***Submit a zip file to Moodle:*** After you have completed all the questions and checked them on Moodle, ***you must submit a zip file with the .cpp file(s) you wrote in Cloud9***. Submit this file going to **Hmwk 2 (File Submission)** on moodle.

Style and Comments (20 points)
Comments (10 points):

- Your code should be well commented. Use comments to explain what you are doing, especially if you have a complex section of code. These comments are intended to

help other developers understand how your code works. These comments should begin with two backslashes (//).

- Please also include a comment at the top of your solution with the following format:

```
// Author: CS1300 Fall 2018
// Recitation: 123 – Your TA
// Cloud9 Workspace Editor Link: https://ide.c9.io/…
// Homework 2 - Problem # ...
```

Algorithm (10 points):

- Remember to include in comments, before the function definition, a description of the algorithm you used for that function.
- This is an example C++ solution to a currency conversion problem.. Look at the code and the algorithm description for an example of what is expected.

```
/**
 * Algorithm: convert the value of USD in Euros.
 *    1. Take the value of number of dollars involved in the transaction.
 *    2. Current value of 1 USD is equal to 0.86 euros
 *    3. Multiply the number of dollars got with the currency exchange rate to
get Euros value
 *    4. Return the new Euro value
 * Input parameters: Amount in USD (double)
 * Output: nothing
 * Returns: Amount in Euros (double)
 */

double convertUSDtoEuros(double dollars)
{
      double exchange_rate = 0.86; //declaration of exchange rate
      double euros = dollars * exchange_rate; //conversion
      return euros; //return the value in euros
}
```

The following algorithm description does not mention in detail what the algorithm does and does not mention what value the function returns. Also, the solution is not commented. This would not receive full credit.

```
/**
 * conversion
 */
double convertUSDtoEurpos(double dollars)
{
      double euros = dollars * 0.86;
      return euros;
}
```

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2

Test Cases (10 points)

Code compiles and runs (**4 points**):

- The zip file you submit to moodle should contain .cpp file(s) that can be compiled and run on Cloud9 with no errors. The functions should match those submitted to the autograder.
- You may create 3 separate files .cpp files with 1 function each, or produce 1 .cpp file with all 3 functions.

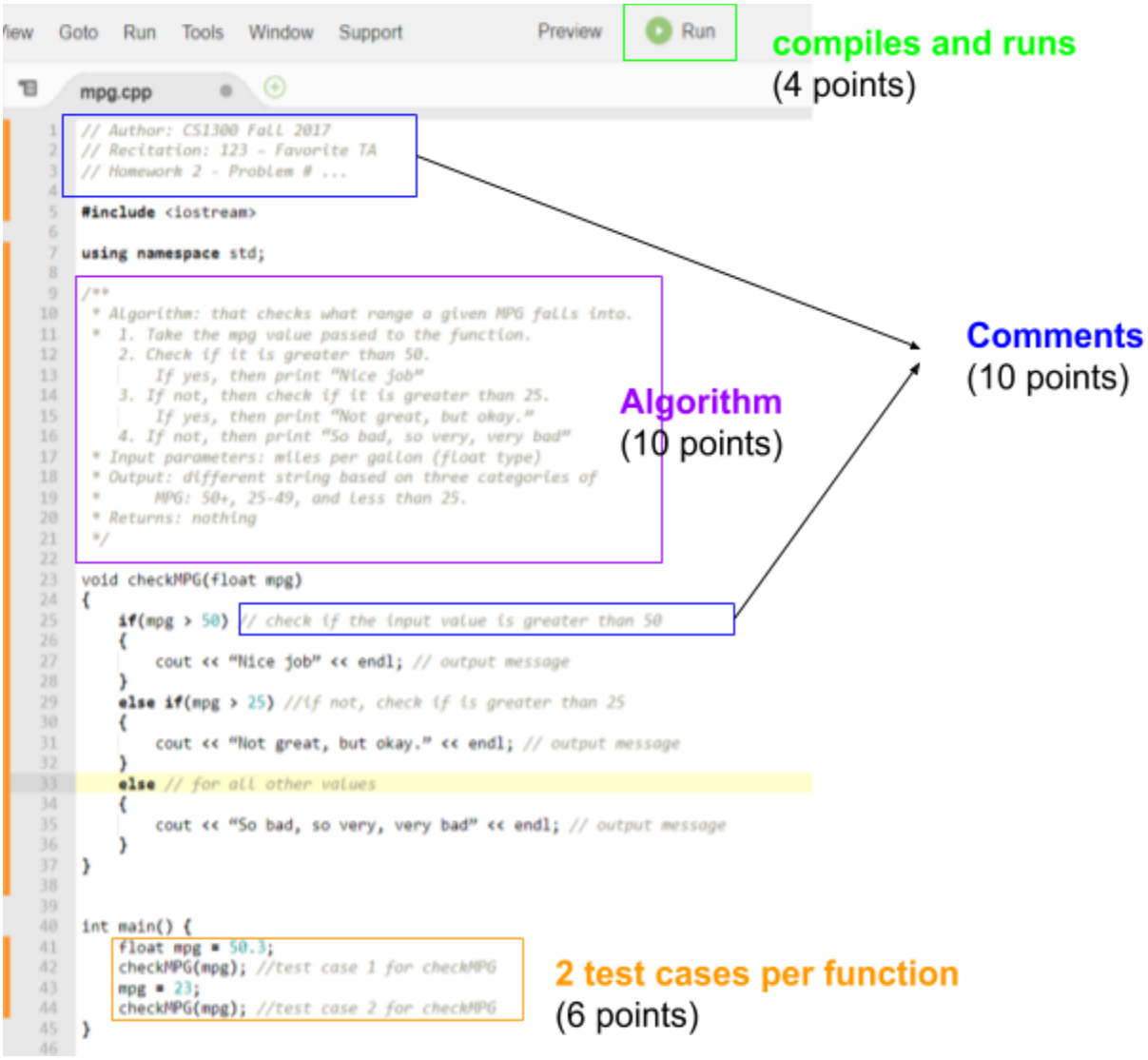
Test cases (**6 points**):

- Each function should have 2 test cases present in your main function(s), for a total of 6 test cases.
- If you choose to create separate files, you will need a main function in each file with the test cases for the function.

```
int main() {
	double dollars = 50.3;
	convertUSDtoEurpos(dollars); //test case 1 for convertUSDtoEuros
	convertUSDtoEurpos(21.5); //test case 2 for convertUSDtoEuros
}
```

The .cpp file(s) you submit should look similar to the example on the next page. Notice that this file has the required comments at the top of the file, an algorithm description, comments within the function, and two tests in main for the function `convertUSDtoEuros`.

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2




```
// Author: CS1300 Fall 2017
// Recitation: 123 - Favorite TA
// Homework 2 - Problem # ...

#include <iostream>

using namespace std;

/**
 * Algorithm: that checks what range a given MPG falls into.
 *  1. Take the mpg value passed to the function.
    2. Check if it is greater than 50.
        If yes, then print "Nice job"
    3. If not, then check if it is greater than 25.
        If yes, then print "Not great, but okay."
    4. If not, then print "So bad, so very, very bad"
 * Input parameters: miles per gallon (float type)
 * Output: different string based on three categories of
 *      MPG: 50+, 25-49, and less than 25.
 * Returns: nothing
 */

void checkMPG(float mpg)
{
    if(mpg > 50) // check if the input value is greater than 50
    {
        cout << "Nice job" << endl; // output message
    }
    else if(mpg > 25) //if not, check if is greater than 25
    {
        cout << "Not great, but okay." << endl; // output message
    }
    else // for all other values
    {
        cout << "So bad, so very, very bad" << endl; // output message
    }
}


int main() {
    float mpg = 50.3;
    checkMPG(mpg); //test case 1 for checkMPG
    mpg = 23;
    checkMPG(mpg); //test case 2 for checkMPG
}
```

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2

Problems Set:
Write a **function** for each of the following problems. You are encouraged to write your solution first in Cloud9. You are required to submit a .cpp file with your functions and a main() function with 2 tests for each of your functions.

Problem 1
Write a function **convertSeconds** that takes one input as seconds and converts it to hours, minutes and seconds.

- Your function **MUST** be named **convertSeconds**
- Your function should accept only one input argument: seconds, as an integer
- Your function should not return any value.
- Your function should print the string in the following format:

  x hour(s) y minute(s) z second(s)

  For example, given input seconds as 3671, it should print :

  1 hour(s) 1 minute(s) 11 second(s)

Problem 2
Write a function to convert a temperature value from Celsius to Fahrenheit and print the temperature in Fahrenheit using the format of the example.

Hint: the formula for converting Celsius(C) to Fahrenheit(F): $F = C*(9/5) + 32$

- Your function name **MUST** be named **celsiusToFahrenheit**
- Your function should accept only one input argument: temperature in Celsius, as an integer
- Your function should not return any value
- Your function should print the resulting temperature Fahrenheit value, with a two digit precision, in the following format:

  For example, given input temperature as 38, it should print :

```
The temperature of 38 in fahrenheit is 100.40
```

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2

Problem 3

The U.S. Census provides information about the current U.S. population as well as approximate rates of change. Using those rates and the current US population, write an algorithm to calculate the U.S. population in exactly one year (365 days). Your function should return the result of your calculations.

Three rates of change are provided:

a. There is a birth every 8 seconds
b. There is a death every 12 seconds
c. There is a new immigrant arriving in the US every 27 seconds

- Your function **MUST** be named **population**
- Your function should accept only one input argument: the initial population, as an integer
- Your function should return the population value in a year.
- Your function should not print/display/output anything

For example, given an initial population of 1,000,000, your function would return **3,482,000**.

Problem 4

In astronomy, **luminosity** is the total amount of energy emitted per unit of time by a star, galaxy, or other astronomical object. We can calculate the luminosity, L, of a star using its apparent brightness, b (how bright it looks from earth), and the star's distance from the observer, d, using the formula:

$$L = 4b\pi d^2$$

Write a function **luminosity** that implements this formula.

- Your function **MUST** be named **luminosity**
- Your function should accept two input arguments: brightness and distance, as doubles
- Your function should return the value of the luminosity as an **int** (you will have to cast it to an integer value)
- Your function should not print/display/output anything
- For **π** use the value 3.14159

For example, given initial values of: b = 1.5 and d = 17.8, your function would return **5972**.

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 2

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.

Remember that you must *also* submit a zip file to moodle with the .cpp file you wrote in Cloud9 to receive full points for this assignment.

What to do if you have questions

There are several ways to get help on assignments in 1300, and depending on your question, some sources are better than others.

- Piazza is a good place to post technical questions, such as how to get user input, or treat that input as an integer. When you answer other students' questions, please do not post your code.
- The Course Assistants (CAs) are also a good source of technical information.
- If, *after reading the assignment write-up*, you need clarification on what you're being asked to do in the assignment, the TAs and the course instructors are better sources of information than Piazza or the CAs.