Assignment #1 – All About Numbers in C++
Due: Monday, 01/20/14, 11:59pm

Grading: For each programming assignment, you are graded by explaining and demoing your code to a TA. You must demo your program within 7 days of turning it in, and if you fail to do so, you will automatically lose 50 points! **Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program**☺ Your grade is based on Polya's 4 steps for solving problems:

- Understanding the problem. (*Recognizing what is asked*.)
- Devising a plan. (*Responding to what is asked*.)
- Carrying out the plan. (*Developing the result of the response*.)
- Looking back. (*Checking. What does the result tell me?*)

(15 pts) Polya's steps 1, 2, and 4 do not directly deal with you C++ code itself, but rather, the steps you took to write a correct program that solves the given problem statement. With this said, make yourself familiar with the three sections below, and make sure you can provide documentation and talk intelligently about what you did to arrive at the solution you did. **You will have to explain how you went through these steps to your TA, as part of your assignment grade!!!**

Understanding the Problem
In your own words, explain what YOU think the problem is asking you to do. In this section, document your uncertainties about the problem and anything else that you feel was unclear or vague. This is to ensure that YOUR understanding matches MY understanding of the problem☺

Devising a Plan/Design
At a minimum, provide an algorithm/pseudo code you designed to help solve the problem. In addition, include pictures/flow charts you used to help you devise your plan, as well as any other design decisions you made such as how to manage your time, how to decompose the problem, where to start first, etc. You can scan any handwritten work and attach it to the document as needed.

Looking Back/Self-Reflection
Report any checking/self-reflection you did while solving the problem. For instance, how did you make sense of the output from the implementation? This includes things such as using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances, using external sources of information such as the internet to make sense of the results, etc. Also, include a statement about what you learned from the assignment.

(65 pts) **Problem Statement:** Write a C++ program that first prints the size of all the primitive types in C++, i.e. int, short, long, float, double, bool, and char, using the sizeof() function. This function returns the number of bytes used by the type supplied to the function. You can use this function by passing the type as an argument, i.e. sizeof(float), sizeof(int), etc.

Now, we want to create variables for the largest and smallest **short**, **int**, and **long**, both **signed** and **unsigned.** You want to assign the max and min of these values to the variables from the **climits** library, i.e. #include<climits>. At this point you probably want to investigate this library by going to http://cplusplus.com. Now, print the values of these values/macros.

Print the values of these variables by adding one to each variable containing the maximum numbers and subtracting one from the variables containing the minimum values (even zero for unsigned). **This should produce overflow, not one number bigger or smaller!!!**

Now, compute the largest and smallest **short**, **int**, and **long**, both **signed** and **unsigned**, using **pow()**, which is in the cmath library and using **sizeof()** for your type. (You do not need to use pow to computer the minimum unsigned number, as that is just zero!). You only need to print the decimal values for these numbers**. Make sure these numbers match the numbers printed from the macros!!!**

Now, we are going to **investigate floats and doubles.** You want to include the **cfloat** library, and **print the number of digits for the mantissa for both types** and the **largest and smallest exponent values for both types**. In addition, let's investigate how we lose precision with floats faster than with doubles. Create a variable for a float and one for a double.

Now, **ask the user to enter a real number for the float, as well as for the double. Subtract .01 from both variables and re-assign the new value**, and then, **print the contents of both variables**. **Let's enter the value .01 for both variables** and see what is printed after subtracting .01. What do you notice?

(10 pts) **Program Style/Comments**
In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!
http://classes.engr.oregonstate.edu/eecs/winter2014/eecs161-001/161_style_guideline.pdf

```
/*************************************************
** Program: numbers.cpp
** Author: Your Name
** Date: 01/17/2014
** Description:
** Input:
** Output:
*************************************************/
```

(10 pts) **Design for Assignment #2**
You will **design a solution** for the following problem statement in Assignment #2. In addition, you will also **provide a test plan** for how you will make sure your program is working correctly. There is design in both the solution and testing your solution.

Problem Statement: Expand the problem from Assignment #1 to find when a float and double seem to fail after subtracting .01 from any number entered by the user. Is there a way we can determine this? How about if we want to determine if a user enters a number too large to be supplied in our variable? How might we determine this? For your design to Assignment #2, you want to:

> Figure out a way to determine how many subtractions/iterations it takes for a float or double to lose precision when subtracting .01 from the users input.
>
> How can we determine if overflow has occurred? How can we make sure a user doesn't enter a number too large to fit into our memory space?

Electronically submit your C++ program (**.cpp** file, not your executable!!!) and design document, **as a pdf**, by the assignment due date, using TEACH.

****NOTE:** The easiest way to upload your program from ENGR to TEACH is to map a network drive to your home directory on ENGR. Mac or Windows, See:
http://engineering.oregonstate.edu/computing/fileaccess/