CS 2100: Data Structures, Spring 2016
Homework 1

Due *via email* by 11:59pm on Saturday, Jan. 21

This is a homework you must complete individually. Each problem here is focused on learning the basic syntax of C++. You should submit a separate cpp file for each problem, commented with your name and the problem number at the top (as well as other comments throughout which describe the program as necessary). Note that these should be tested before submitting them; make sure they actually compile and work!

Please type all answers and email to the me at echambe5 - at - slu.edu by 11:59pm on the date due. Please also cc the grader for this one - mmeyer71 - at - slu.edu.

1. Write a short C++ program that prompts the user to input a double value x and then computes and prints the number of times we can divide x by 2 before we get a number less than 2.

2. Write a short C++ program that prompts the user to enter an integer n, and then accepts n floating point numbers as input. Your program should then print the maximum, minimum, and average of the list of numbers. For example, if the user enters 5, followed by 3.2, 1.1, 10, 12.5, and 23, the output should be:

```
minimum: 1.1
maximum: 23
average: 9.96
```

3. A number n is *perfect* if it is equal to the sum of its proper divisors, where a proper divisor is a number which divides n and is less than n. For example, 6 is a perfect number because $1 + 2 + 3 = 6$, and 1, 2, and 3 are the only divisors less than 6. Similarly, 28 is perfect also, because $1 + 2 + 4 + 7 + 14 = 28$.

   Write a function isPerfect that takes an input n as input and returns the boolean true if n is perfect, and false otherwise. Then test your function by writing a main program that checks for all perfect numbers in the range 1 to 9999 by testing each number one at a time. When a number is found to be perfect, your program should print it to the screen.

   Note: The first two lines of output should be 6 and 28, and your program should find two other perfect numbers as well.

4. Extra Credit:

   Write an efficient C++ function that takes any integer value i and returns 2^i as a long value. Your function should *not* multiply 2 by itself i times; there are much faster ways of computing 2^i.

   Note: Faster solutions will receive more extra credit.