CS 180: Data Structures, Fall 2012
Homework 1

Due *via email* by 11:59pm on Saturday, Sept. 8

This homework covers the transition guide and Ch1 of the text. Please read and familiarize yourself with this material before working on the homework. Note that these should be tested before submitting them; make sure they actually compile and work!

Please type all answers and email to the instructor at echambe5 - at - slu.edu by 11:59pm on the date due.

1. Write a short C++ program that prompts the user to enter an integer n, and then prints a "staircase" of size n. For example, if the user enters the number 5, the following will be printed:

```
    *
   **
  ***
 ****
*****
```

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: Problem C-1.10 from the text:

   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.