Name: ____________________________________

A Good `/* comment */` is Worth a Thousand Lines of Code
ASSERTions, INVariants and PRE-POST-Conditions

(0) From the onCourse site and fetch a "Starter Kit" for Lab6. **Unzip** this on your Desktop.
(1) Create a new CONSOLE project.

From the `INVariant` folder, place the C++ lines in **inv1.cpp** into your main() function. **Complete the code that is necessary in order to make the INVariant and ASSERTion true.**

-------------------------- **Call me over when you have this working.** ---------------------------

(3) Modify `inv1.cpp` to use other values for N. On paper, calculate by hand what you expect for output for the following values of N. **Show me your algebra and arithmetic not just your answer that will be output. Circle the final sum.**

N	sum (final value)
10	
100	

(4) Complete a more thorough **white-box test** of this code by arriving at a solid set of **boundary value** test values; fill in your boundary values to test below. Test your code on these values.

Note: rather than continually change the value of the constant N with the editor, prompt the user and read in various values from the keyboard (stdin) to try.

N	sum (final value)

(5) What is the largest value of N you can use without causing an overflow? Rather than do this by brute force and testing of *lots* of test cases, can you **algebraically solve** for the largest value of N that would work before the memory overflows? Hint: LONG_MAX, INT_MAX, and **`SHRT_MAX`** are built-in constants (`#include <limits>`) that hold the largest possible values allowed in the respective variables of those types. Show your work below.

(6) Assuming you find the value of N in #5, where would you place an **`assert()`** into your code to make it "safe"?

-------------------------- **Call me over when you have this working.** ---------------------------

(7) From the INVariant folder, load the code from the file **`inv2.cpp`** and complete the code that is necessary in order to make the INVariant and ASSERTion true.

-------------------------- **Call me over when you have this working.** ---------------------------

OOP – Object-oriented Programming

(8) Start a new **medPing** project (or look in your Lab5 or Program #a2 Project).

(a) Open **medPing.h** and **medPing.cpp**
(b) What **private methods** does a medPing object have?

(c) What **private data members** does a medPing object have?

(d) Look in **mpPatient.h**. What is this?