Programming Assignment 1
Out: September 15, 2015, Tuesday -- DUE: October 1st, 2015, Thursday, 11:59pm
EC327 Introduction to Software Engineering – Fall 2015

Total: 100 points

- *You may use any development environment you wish, as long as it is ANSI C++ compatible. Please make sure your code compiles and runs properly under the Linux environment on the PHO307/305 (or eng-grid) machines before submitting.*
- *Labs may be submitted up to a week late at the cost of a **30% fixed penalty** (e.g., submitting a day late and a week late is equivalent). It is in your best interest to complete as many lab questions as possible before the deadline. If you have missing questions in your original submission, you may complete and submit the missing solutions during the following week. Any submissions after the deadline will be subject to the 30% penalty. No credit will be given to solutions submitted after the 1-week late submission period following the deadline.*
- *Please make sure to follow the assignment submission guidelines in this document not to lose points.*

Please write C++ code for solving the following problems related to data types, operations, if statements, simple loops, and basic functions (Chapters 2-3-4-5 of the textbook by Liang).

Q1: *Perimeter and area of triangles.* [20 points]
Write a program that reads in the three sides of a triangle from the console and checks whether the given dimensions forms a triangle. If the three given side-lengths constitute a triangle, then your code should print the area and perimeter on the console. Otherwise, the program prints a message.
Hint: Heron's formula provides a method to compute a triangle's area given the lengths of all three sides.

Text in < > in the examples demonstrates the user inputs entered via the keyboard.

Sample runs:

```
Enter the dimensions of the triangle:
Side-1: <10> <enter>
Side-2: <2> <enter>
Side-3: <3> <enter>
Dimensions 10, 2, 3 do not form a triangle. [Program Exits]

Enter the dimensions of the triangle:
Side-1: <5> <enter>
Side-2: <4> <enter>
Side-3: <3> <enter>
This triangle's area is 6 square-units and its perimeter is 12 units. [Program Exits]

Enter the dimensions of the triangle:
Side-1: <10> <enter>
Side-2: <23> <enter>
Side-3: <24> <enter>
This triangle's area is 114.23* square-units and its perimeter is 57 units. [Program Exits]
```

**You should display at least two fractional digits. You can display more of the fractional part if you like.*

Q2. ***Converting Units*** **[10 points]**

Write a program that converts numbers among Celsius, Fahrenheit, and Kelvin. Note that your program should be capable of doing the conversion from any of these to the other. The program should start by asking the user to select the conversion type. **If an incorrect code is entered (e.g. 6 or -1) an error message should be shown and the user should be prompted again.**

```
Sample run:

Celsius to Fahrenheit (enter 0)
Celsius to Kelvin (enter 1)
Fahrenheit to Celsius (enter 2)
Fahrenheit to Kelvin (enter 3)
Kelvin to Celsius (enter 4)
Kelvin to Fahrenheit (enter 5)
Conversion type:  <User enters 0-5. Assume 1 for this example.><enter>
Enter the amount in Celsius: 10<enter>
10 Celsius is 283.150 Kelvin*. [Program Exits]
```

**You should display at least three fractional digits. You can display more of the fractional part if you like.*

Q3. ***Hamming Distance*** **[25 points]**

In information theory, the **Hamming distance** between two sequences of equal length is the number of positions at which the corresponding symbols are different. Put another way, it measures the minimum number of *substitutions* required to change one string into the other, or the number of *errors* that transform one string into the other.

Examples:

The Hamming distance between:

- "**toned**" and "**roses**" is 3.
- **1011101** and **1001001** is 2.
- **2173896** and **2233796** is 3.

(Description above is from Wikipedia: http://en.wikipedia.org/wiki/Hamming_distance)

Write a program that prompts the user to enter two positive integers (in decimal; up to 32-bits of precision) and computes the Hamming distance between the two numbers when the numbers are represented in base-16 (hex format). The program then displays the Hamming distance on the screen.

```
Sample runs:

Enter two positive integers:  <145363><enter>
<54637823><enter>
Hamming distance between 145363 and 54637823 when numbers are in hex format is: 7    [Program
Exits]*

Enter two positive integers:  <262178770><enter>
<262637534><enter>
Hamming distance between 262178770 and 262637534 when numbers are in hex format is 2. [Program
Exits]**
```

** 145363 and 54637823 are represented as 237D3 and 341B4FF, respectively, in hex form.*

***262178770 and 262637534 are represented as FA087D2 and FA787DE in hex form.*

Q4. *Letter Conversion* [25 points]

Write code that asks the user to enter a character. Then the program asks the user to enter a non-negative integer offset. This program adds the offset to the character to produce a new ASCII value. This value should then be displayed. If the offset is 0 and the character is a letter, the program should change the case of the letter. *("Change case" means that a lowercase letter should be converted to uppercase, and an uppercase letter should be changed to lowercase.*) If the offset is 0 but the character is not a letter, the same character should be displayed.

Sample runs:

```
Enter character: <D>
Offset (enter 0 to convert case): <7>
New character: K

Enter character: <a>
Offset (enter 0 to convert case): <0>
New character: A

Enter character: <B>
Offset (enter 0 to convert case): <0>
New character: b

Enter character: <3>
Offset (enter 0 to convert case): <0>
New character: 3
```

Hint: In the ASCII table (see Appendix B of your book), uppercase letters appear before lowercase letters. The offset between any uppercase letter and its corresponding lowercase letter is the same. You can compute this offset by:

int offset = 'a' – 'A';

or by: int offset = 'A' – 'a';

Notice that your program should error check if the resulting char code (original code + offset) is greater than X (you need to figure out X). You'll notice in the ASCII table that this does not make sense. Think about why X is the limit.

Q5. ***Warmer or Colder?*** **[20 points]**
Write a program to help you guess an <u>integer</u> number. Initially the program should create a random number (i.e. the answer). Then the user should input a guess. After the first guess, the computer should say whether the new guess is warmer (closer) or colder (farther) from **the last guess** (absolute value of the difference). In the event that the distance is the same it should say "No change".

Sample runs:

```
Enter your first guess: <100> <enter>
Enter your next guess: <90> <enter>
Colder
Enter your next guess: <170> <enter>
Colder
Enter your next guess: <110> <enter>
Warmer
Enter your next guess: <120> <enter>
Correct! The number was 120!
[Program Exits]

Enter your first guess: <100> <enter>
Enter your next guess: <90> <enter>
Warmer
Enter your next guess: <50> <enter>
No change
Enter your next guess: <70> <enter>
Correct! The number was 70!
[Program Exits]
```

**Hint: http://www.cplusplus.com/reference/cstdlib/rand/*

Submission

Please use the file names **Q1.cpp, Q2.cpp, etc.** for questions 1-5, respectively. Put all your .cpp files in a folder named <username>_PA1 (e.g., dougd_PA1), zip it, and submit a single file (e.g., dougd_PA1.zip). Upload the files to the **BLACKBOARD** submission site by 11:59pm on the due date. Do **NOT** submit your executable files (a.out or others) or any other files in the folder. Make sure to **comment** your code.

In addition to this primary submission mechanism, there will be an additional submission requirement AFTER the due date. This will NOT require additional coding on your part. This requirement will familiarize you with versioning systems. We will provide additional information on this requirement at a later date.