Programming Assignment 1
Out: September 20th, 2017, Wednesday -- DUE: October 6th, 2017, Friday, 11:59pm
EC327 Introduction to Software Engineering – Fall 2017

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.*
- *PAs 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 PA 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: *Intersecting Rectangles* [20 points]
Write a program that reads in two rectangles from the console (STDIN). The rectangles are indicated by their (x, y) lower left coordinate and their length (x) and height (y). You do NOT need to deal with negative numbers and can assume all inputs are integers. The program should then tell if the rectangles intersect or not. You are able to use the <math.h> library in this assignment. For more see:
http://www.cplusplus.com/reference/cmath/

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

Sample run:
Enter the information for the first rectangle
x-coordinate: <2> <enter>
y-coordinate: <0> <enter>
length: <4> <enter>
height: <5> <enter>
Enter the information for the second rectangle
x-coordinate: <8> <enter>
y-coordinate: <3> <enter>
length: <9> <enter>
height: <1> <enter>
THE RECTANGLES DO NOT INTERSECT [Program Exits]

Enter the information for the first rectangle
x-coordinate: <2> <enter>
y-coordinate: <4> <enter>
length: <5> <enter>
height: <6> <enter>
Enter the information for the second rectangle
x-coordinate: <1> <enter>
y-coordinate: <2> <enter>
length: <5> <enter>
height: <4> <enter>
THE RECTANGLES INTERSECT [Program Exits]

Q2. *Cooking converter* [10 points]

Write a program that converts numbers among teaspoons, tablespoons, ounces, and cups. 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. 5 or -1) an error message should be shown and the user should be prompted again.**

Sample run:
Cups = 1, Ounces = 2, Tablespoons = 3, Teaspoons = 4
Conversion from: <User enters 1-4. Assume 1 for this example.><enter>
Conversion to: <User enters 1-4. Assume 2 for this example.><enter>
Enter the amount in Cups: 10<enter>
10 Cups is 80 Ounces*. [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. *Rock, Paper, Scissors* [20 points]

Write a program to play "rock, paper, scissors". In this game, you first prompt the user to choose "rock, paper, or scissors". After this, the computer should randomly generate one of these three. Then you should display the choice of the computer and determine the winner (rock beats scissors, scissors beat paper, paper beats rock).

Sample run:
Choose Rock (0), Paper(1), or Scissors (2): *<1> <enter>*
Computer chooses: Rock
You win!
[Program Exits]

Choose Rock (0), Paper(1), or Scissors (2): *<2> <enter>*
Computer chooses: Rock
You lose!
[Program Exits]

Choose Rock (0), Paper(1), or Scissors (2): *<0> <enter>*
Computer chooses: Rock
You tie!
[Program Exits]

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

Q5. *Simple Ciphering with ASCII* [25 points]

Write code that asks the user to enter a single letter. Then choose one of three operations: (1) change case; (2) reverse alphabet; (3) encrypt.

1. Change case should convert the letter to the opposite case (upper to lower; lower to upper).
2. Reverse alphabet should find the corresponding letter starting from the end of the alphabet AND change the case. For example, 'a' equals 'Z' and 'Y' equals 'b'.
3. Encrypt should take a letter and change it six letters "ahead" (assume the alphabet "wraps around" and all of the same case letters are together). For example 'C' is 'I' and 'z' is 'F'

If the input is not a letter (e.g., user entered a number instead of a letter), your code should print an error message. If the input is more than one letter, an error message should also be produced.

Sample runs:
Enter char, operation:*<b> <1><enter>*
Result: B [Program Exits]

Enter char, operation:*<E> <2><enter>*
Result: v [Program Exits]

Enter char, operation:*<K> <3><enter>*
Result: Q [Program Exits]

Enter char, operation:*<*> <1><enter>*
Result: Illegal char [Program Exits]

Enter char, operation:*<g> <5><enter>*
Result: Illegal operation [Program Exits]

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.