UF UNIVERSITY of FLORIDA
Herbert Wertheim College of Engineering
Spring Semester 2015

COP 3503

HW1 – Intro to C++

Overview

This homework assignment's purpose is to get you familiar with C++ and to start digging into its basic constructs. The specification for these assignments is taken from the Java-based variant you may have seen in earlier coursework. For equity, the Java-variant solutions are included within the assignments materials. Project data are included for the CLion IDE. Tests for each project are also included and can be run from within the development environment. Files to be submitted are as follows:

Quadratic Root Tool:	**`QuadraticRootTool.cpp`**
Grade Calculator:	**`GradeCalculator.cpp`**
Credit Card Validator:	**`CreditCardValidator.cpp`**

Each program ***must*** be in the file specified above and have its own distinct **`main()`** function. (They should be standalone programs.) If you use Eclipse, each will need to be in its own project to prevent errors (due to multiple **`main()`** functions.)

Objectives

Upon completion of this activity, students should be able to...

- Design, build, and execute a simple C++ program
- Use basic data types and I/O functions to construct a text-based application
- Create and utilize stand-alone, procedural functions

Instructions

This assignment is composed of three separate programs that will be graded independently of one another.

WARNING: Your output must match the examples EXACTLY to receive credit. Deviations will result in reduction of score on the assignment. Thoroughly test your code to be sure it compiles and executes properly according to standard C++! **Submissions that do not compile will be marked zero.**

Quadratic Root Tool

The roots of a quadratic equation in the form $\boldsymbol{ax^2 + bx + c = 0}$ can be obtained using these formulas:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad and \quad r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$\boldsymbol{b^2 - 4ac}$ is called the discriminant of the quadratic equation. If it is **positive**, the equation has two real roots; if it is **zero**, the equation has one root. If it is **negative**, the equation has no real roots.

Write a program that prompts the user to enter values for *a*, *b*, and *c* and displays the result based on the discriminant. If the discriminant is positive, display two roots. If the discriminant is 0, display one root. Otherwise, display "The equation has no real roots."

Note that you can use **`sqrt(x)`** (***`#include <cmath>`***) to compute $\sqrt{x}$. Here are some sample runs; please match this output format EXACTLY (matching at least 4 decimal places):

```
Enter a, b, c: 1.0 3 1↵
The equation has two roots: -0.38166 and -2.61803
```

```
Enter a, b, c: 1 2.0 1↵
The equation has one root: -1
```

```
Enter a, b, c: 1 2 3↵
The equation has no real roots.
```

Grade Calculator

Write a program that reads student scores, gets the best score (**`BestScore`**), and then assigns letter grades based on the following scheme:

A: [BestScore, BestScore – 10]
B: (BestScore – 10, BestScore – 20]
C: (BestScore – 20, BestScore – 30]
D: (BestScore – 30, BestScore – 40]
F: Lower than BestScore – 40.

The program should prompt the user to enter the total number of students, then prompt the user to enter all of the scores, and conclude by displaying the letter grades. Here is a sample run:

```
Enter the number of students: 4↵
Enter 4 scores: 40 55 70 58↵
Student 0 - Score: 40, Letter: C
Student 1 - Score: 55, Letter: B
Student 2 - Score: 70, Letter: A
Student 3 - Score: 58, Letter: B
```

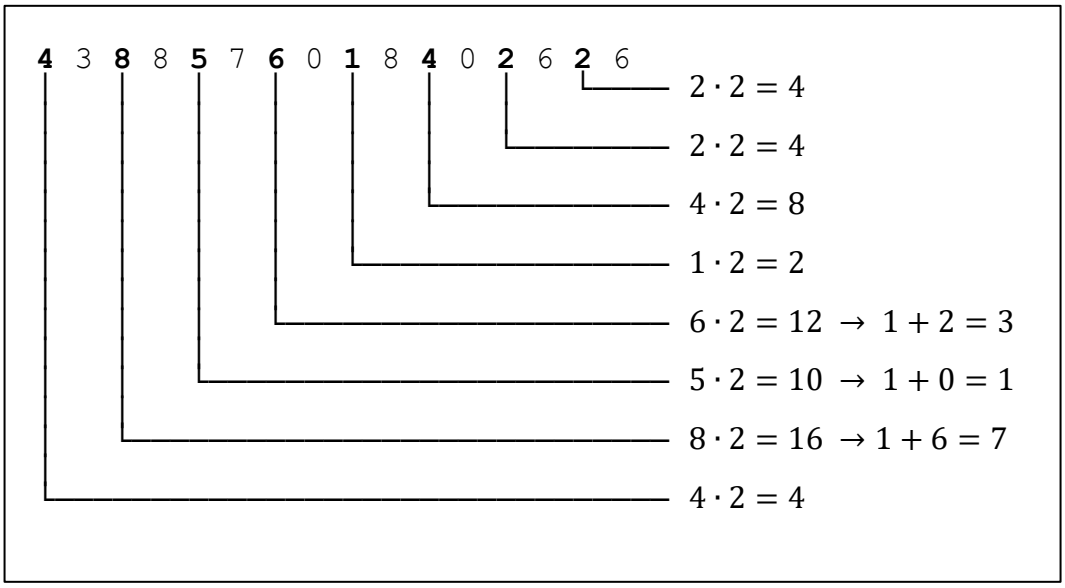
Credit Card Validator

Credit card numbers follow certain patterns. A credit card number must have between 13 and 16 digits. In addition, cards have prefixes by type:

4 – Visa
5 – Mastercard
37 – American Express
6 – Discover

In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine whether a card is entered correctly or whether a credit card is scanned correctly by a scanner. Credit card numbers are generated following this validity check, commonly known as the Luhn check. The Luhn check can be described as follows. Consider the card number 4388576018402626:

1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the digits to get a single-digit number.



2. Now add all single-digit numbers from Step 1.

$$4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37$$

3. Add all digits in the odd places from right to left in the card number.

$$6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38$$

4. Add the results from the previous two steps.

$$37 + 38 = 75$$

5. If the result of the addition is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

Write a program that prompts the user to enter a credit card number. Display whether the number is valid or invalid. Design your program to use the following functions:

```
// Return true if the card number is valid.
bool isValid(long long number)

// Get the result of Step 2.
int sumOfDoubleEvenPlace(long long number)

// Return this number if it is a single digit; otherwise, return the sum
// of the two digits.
int getDigit(int number)

// Return sum of odd-place digits in number.
int sumOfOddPlace(long long number)

// Return true if the digit is a prefix for this number.
bool prefixMatched(long long number, int digit)

// Return the number of digits in number
int getSize(long long number)

// Return the first numDigits digits from number. If the no. of digits
// in number is less than numDigits, return number.
long getPrefix(long long number, int numDigits)
```

Here are example runs of the program:

```
Enter a credit card number: 4388576018410707 ↵
4388576018410707 is valid.
```

```
Enter a credit card number: 4388576018402626 ↵
4388576018402626 is invalid.
```

Submission

Your code and any accompanying documentation should be submitted online through the Canvas website's Assignment tab for this homework. All submission data must be submitted within the root of the zip file. ***You must include the source (cpp) files in your submission.***

Submissions that do not follow this specification may be marked zero (no credit).