CSCE 121(505-508) **Homework 2**
Due 11:59 p.m., 2/3/14 [140 points]

This Homework assignment is meant to give you some experience writing C++ programs that use some of the fundamental operations: loops, comparisons, functions, and vectors. Also, it is meant to give you an opportunity to write code that catches errors and throws exceptions.

All work should be turned into CSNet by the deadline. **In addition, please bring a hardcopy printout of your code to class the next day.** For programs, you need to turn in only the source code (not object or executable code). Your code will be tested using g++: you are welcome to develop in Visual Studio, but please make sure your code also runs in g++.

0) Create a text file, README, in which you:
    a. State the Aggie Honor statement, or else explain why you cannot do so.
    b. List any resources used, outside the textbook and discussions with the Instructor, TA, or Peer Teacher
    c. List any known problems with the assignments you are turning in. For example, if you know your code does not run correctly, state that. This does not need to be a long explanation.
    d. For places where you may have done some additional work (e.g. in helping the computer make its choice in the game), put in a brief summary detailing what you did.
1) [80 points] Write a program to play the game Rock-Paper-Scissors-Lizard-Spock. This game is an extension of Rock-Paper-Scissors, a 2-player game where each player simultaneously gives a hand signal with one of the three choices. If both players make the same choice, the game is a tie, and they play again. If the choices are different, one player beats the other, depending on the combination. The rules for which side wins can be found on the Wikipedia page (and elsewhere): http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock
    a. For a game, let the user enter one of the five choices. The user should be able to either type the whole name out, or else use a letter (r for rock, p for paper, s for scissors, l for lizard, or k for spock). You should handle invalid input smoothly, so that if a user mistypes, the program does not crash! [20 points for handling input]
    b. After the user enters their choice, the computer should print out its choice (made independent of the user), along with a message indicating who won. For full credit on this part, you should be clearly descriptive in the output. [15 points for output, 15 points for determining winner correctly]
    c. The computer's choice should be chosen in a way that at least appears to be random. As an initial approach, just create a vector with a bunch of preprogrammed choices that the computer makes, but will repeat. For full credit, improve on this: look up how to make the choices change over time, or be more random. [20 points]
    d. After each game, keep and print out a running total of how many games each player (human and computer) has won. Play the game repeatedly, until the user enters "quit" or "q". [10 points]
    e. Some hints:
        i. Consider using the try-catch to catch input errors (though you can also just use if statements to get the same result).
        ii. Write a function that can handle the comparison to find a winner, to separate that code from the input/game loop.
2) [60 points] Write a program to compute compound interest, as follows:
    a. Write a function that takes in three values: a starting balance (floating point), an integer number of periods (integer), and an interest rate (floating point) expressed as a percentage. It should return a floating point value that is the ending balance. [15 points]
        i. Note: each period, the balance should increase by the percentage given, and this should compound. For instance, if the function took in the parameters 1000.0, 3, 2.0, this means a starting balance of 1000 that increases by 2% for 3 periods. After 1 period the balance would be 1020.0, after 2 it would be 1040.40, and after 3 it would be 1061.208 (and this would be returned).
    b. Make sure your function checks for basic errors and throws the appropriate exceptions. [10 points]
    c. Your program should let the user repeatedly enter a set of values to compute (i.e. a balance, interest rate, and number of periods). [20 points]
        i. Keep reading data until the user enters a balance of 0.
        ii. Store the values read in into vectors.

iii. Be sure to catch any bad input.

d. Once all data has been entered, then compute the interest for each case, and output all results to the screen in an organized fashion. [15 points]
e. Remember: you should always use good descriptions for requesting input and formatting output.