CSCI-1200 Computer Science II — Fall 2006
Homework 5 — Superhero Friends

This assignment is a fun (& silly) departure from the previous assignments that focused on the practical use of complex STL data structures and iterators. You will create two new classes that represent individual superheroes (`Superhero`) and teams of superheroes (`Team`). We provide sample code in the `main.cpp` file demonstrating how these classes should behave and interact. This assignment will give you lots of practice overloading operators and provide an example where creating a friend relationship between classes is useful. *Please carefully read the entire assignment before beginning your implementation.*

The program does not require any input or command line arguments. The provided test code in `main.cpp` uses `assert()` to verify that your classes are declared and implemented correctly. One of your tasks is to write additional test cases using this same strategy. If all test cases pass, a simple message is output to the screen and the program exits normally.

First create the `Superhero` class that stores the hero's name, their true identity, and their superhuman power, all as strings. You should write read-only accessor functions to get the hero's name and power. However, it is important that each superhero's true identity remain a secret, so *it must not be accessible through the public interface*. The only way to discover a superhero's true identity is to correctly guess their true identity by using the `operator==` and `operator!=` functions.

You will read in data through the input stream (`>>`) operator. Each line of input contains the three strings necessary to initialize a new superhero. Practice using the `getline` and `substr` functions when you implement this operator. You will also implement the output stream (`<<`) operator. Study the provided test code for examples of proper I/O. The test code for the input and output stream functions uses STL `stringstream`s, but these operators will work with standard I/O (`cin` and `cout`) or file I/O (`ifstream` and `ofstream`) as well.

In the sample code we use the unary `operator-` to negate (a.k.a. corrupt) a superhero. Superheroes are initially good, but turn to evil if corrupted. Likewise, the operation can be applied in reverse to turn an evil *supervillain* into a good superhero. Another fun example is `operator>` that can be used to settle debates about which hero's superpower is greater. Our test cases do not fully specify the differences in greatness for all powers. Your task is to define a ranking system for the remaining powers. However, it is important to note that this property is not necessarily *transitive*; that is, if `a > b` and `b > c`, it *does not necessarily hold that* `a > c`.

The final component of the assignment is to implement teams of superheroes. Various versions of `operator+`, `operator+=`, and `operator-=` are used in the example code to build, add to, remove from, and merge teams. Think carefully about the parameter and return types of each of these operators. The `Team` class includes an accessor function to get the name of the team. This name is automatically formed by taking the first consonant of the first team member's true identity, the first vowel of the first team member's true identity, the first consonant of the second team member's true identity, the first vowel of the second team member's true identity, etc. You will need to use a friend relationship to correctly implement this function.

For extra credit, be creative and extend the functionality of the superhero program. Select an operator that has not yet been overloaded and prepare a story behind how it is used on superheroes or teams of superheroes. Implement the operator and create appropriate test cases. In your `README.txt` file, describe your operator and justify why it is an *intuitive* command appropriate for operator overloading.

Do all of your work in a new folder named `hw5` inside of your CSII homeworks directory. Use `const` and pass/return by reference when appropriate. Please use the provided template `README.txt` file for any notes you want the grader to read. **You must do this assignment on your own, as described in "Academic Integrity for Homework" handout. If you did discuss the problem or error messages, etc. with anyone, please list their names in your `README.txt` file.** When you are finished please zip up your folder exactly as instructed for the previous assignments and submit it through the course webpage.