CSCI-1200 Data Structures — Fall 2025
Lab 8 — Operator Overloading & Superhero Friends

Checkpoint 1

estimate: 30-50 minutes

Checkpoint 1 will be available at the start of Wednesday's lab.

Checkpoint 2

estimate: 15-25 minutes

This lab is a fun (& silly) departure from our focus on the practical use and implementation 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 lab will give you practice overloading operators.

http://www.cs.rpi.edu/academics/courses/fall25/csci1200/labs/08_operators/main.cpp
http://www.cs.rpi.edu/academics/courses/fall25/csci1200/labs/08_operators/team.h
http://www.cs.rpi.edu/academics/courses/fall25/csci1200/labs/08_operators/team.cpp

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. See the examples in the `main.cpp` file. Complete the necessary implementation so that the code compiles and runs successfully (with no assertion failures).

To complete this checkpoint: Show one of the TAs your implementation.

Checkpoint 3

estimate: 15-25 minutes

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 and implement 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`.

To complete this checkpoint: Show a TA these additions and the test output.

Checkpoint 4 – *EXTRA CREDIT*

estimate: 15-25 minutes

The final component of the lab is to finish the implementation for 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. Study carefully the provided prototypes and the parameter and return types of each of these operators. Your job is to implement these operators. We provide the implementation of a `Team` member 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 add a friend relationship to compile this code.

Finally, be creative and select an operator that has not yet been overloaded and prepare a story behind how it is used on superheroes or teams of superheroes. You do not need to implement this operator.

To complete this checkpoint and the entire lab: Show a TA your completed program and discuss your proposed new operator. NOTE: You can only receive this extra credit if you have completed Checkpoint 1.