Post Mortem 1
CS247 S17

Post Mortems are released after each assignment as a way of detailing the common errors that were found by the markers. By reviewing these errors, as well as the individual feedback for your assignment, you can learn which areas you can improve on for future assignments/exams.

General:
• Comments should be used to help the reader of your code understand what your code is doing. While ideally your variable/function names should give a clear idea of how your code works, if you write a particularly obscure or complicated section of code, a meaningful comment should be made to assist the reader. A comment like “This is the mutator” is neither meaningful nor necessary.

• Be aware of how long your lines are. If you writing beyond 80 characters on a single line, consider breaking that line up to improve readability. This problem was particularly present in the text files submitted for question 1.

• Data members should not be public. Users of your classes should only be able to view and change data members through mutators and accessors. This is so you can explicitly control how users can interact with your class.

Question 1:
• A default constructor is a constructor that takes no parameters. A compiler-generated constructor is a default constructor that is automatically provided to you if you do not define your own. There were some students who did not understand the distinction between these two terms.

• An object being mutable does not guarantee that it will also be entity-based. It is possible (but rare) to have mutable value-based ADTs.

• Many students said that a Collection was an entity-based ADT. While we did accept this answer if it was well justified, the intent was for a Collection to be value-based. While the Collections refer to real-world User entities, the Collections themselves are just a list of those entities. As well, while a Collection could be a list of users, which would lead towards being entity-based, you could have multiple Collections with overlapping values.

Question 2:
• Some students didn’t abstract common data members (i.e. _email_address and _id) in base class Account; some students didn’t call Account::validateJSON in subclass validateJSON which led to duplicate code.

• Some students forgot the keyword “override” in subclass methods. While it’s not required, it’s useful for debugging.

• It is useful to remember that base class methods can still be used by derived classes using the appropriate scope operator.

**Question 3:**

• If your code is messy and overly complicated, it is likely you didn’t plan out the details of your program before you started coding. Aim to keep your program as simple and straightforward as possible to avoid needing to implement convoluted solutions to get your code to run.

• Collection::~Collection() is responsible for deleting both each user in the collection and the entire collection.

• The copy operations in Graph should do deep copies, and the equality operator should do a deep equality check, but it would be reasonable to share (User *).