CSC324 Assignment 3 Marking Scheme

Read through this carefully before submitting - the last thing we want is for you to lose marks because you didn't know what we would be looking for! Please note for the autotesting: we have given you an INCOMPLETE set of module export names. Make sure you add in all of them as you finish them, so that you don't accidentally forget to export something and have your program fail to compile!!

AList Correctness: 10%

Your mark in this section will be determined solely on the percentage of tests passed for the `AList` module.

Mutation Correctness: 40%

Your mark in this section will be determined solely on the percentage of tests passed for the `Mutation` module.

Mutation User Correctness: 10%

Your mark in this section will be determined solely on the percentage of tests passed for the `MutationUser` module.

Design (Parts 1-4): 10%

We are looking for a few different style and design details here:

- Are your functions modular? No function should be very long (this is especially true in Haskell) or complex; make use of helper functions liberally.
- Make sure you haven't imported any external modules. Exception: if you want to make `StateOp` an instance of `Monad` (not required), you may import `Control.Applicative` to also make it an instance of `Applicative` and `Functor`, to get your program to compile.
- Make use of higher-order functions when possible; if using recursion, use pattern-matching when possible.

Compound Mutation (Part 5): 15%

We will not be autotesting your work in CompoundMutation; instead, we will be reading it carefully to understand your design. We are looking for a few different things:

- Have you correctly added a `Pointer` constructor, and not added a `Value` constructor?
- Have you correctly made `Person` an instance of the `Mutable` type class?
- Are you making use of the "integer pointer" and "boolean pointer" primitives in your solution?
- Have you correctly implemented `@@, age`, and `isStudent`? You are not strictly required to make age and `isStudent` functions, but do keep in mind that for the best learning experience, you should think about how to make `@@` a polymorphic as possible.

Documentation and Type Signatures: 15%

All of your functions should be well-documented (in English). You do not need to follow a standard format, but make sure the purpose of each function is clear. All of your functions must have type signatures, except you shouldn't display type signatures when implementing functions for an instance of a type class (these type signatures are defined in the type class itself). Remember that type signatures are part of good documentation - looking at types alone often gives a great deal of insight into the nature of the function.
10% of the grade is for Parts 1-4 (your work in `AList.hs`, `Mutation.hs`, and `MutationUser.hs`), and 5% is for Part 5 (your work in `CompoundMutation.hs`).