Week 5 Lab: Types and Type Systems

In this lab, we will be working with our own types and familiarizing ourselves with the Maybe (introduced in lecture) and Either type constructors using our own user defined types.

Task 1: Working with Account and Post Types (10 Points)

In this lab, we have provided you the following definitions of `Account` and `Post`:

```
data Account = Account String String (Maybe String) [Account] [Post] deriving (Show, Eq)
data Post = Post Int (Either String Int) deriving (Show, Eq)
```

An `Account` has a username, email address, potentially a phone number, a list of accounts that it is following, and a list of posts.

A `Post` has a numeric identifier and a type that indicates if comments are disabled and if not, provides the number of comments on the post.

Step 1: Write the function `postsWithMinimumComments` that takes a list of posts, a minimum comment count, and returns posts within that list that have comments enabled and have more comments than the minimum comment count.

Step 2: Write the function `removePhoneNumbers` that takes a list of accounts and removes all phone numbers from users who have provided one.

Step 3: Write the function `followingBack` that takes an account A, a list of accounts, and returns a list of accounts that are following account A back. It might be useful to know that since we are deriving Eq in both of our types, you are able to compare Account instances directly using (==).

Step 4: Write the function `retrieveAllCommentedPosts` that takes a list of accounts, and returns all commented-posts from all accounts in one flattened list.

Task 2: A2 Warmup (Optional)

Assignment 2 should be released now and there is a warmup task provided that will be extremely useful. If time persists after you have completed this week's lab, please begin the Assignment 2 warmup!

Submission and Instructions

Submit the files `W05lab.hs` on Markus. To give you faster feedback on labs, you will be able to test your own code on Markus once every hour. However, it is up to you to debug and determine why your code pass or fail tests. On labs, the tests that you are able to run on Markus are the same tests that we will be running to determine your grade.

Please test your labs early. Markus sometimes hangs when running tests, particularly if you submit code that does not terminate. If so, you will not be able to continue to submit additional tests until the test times out.

For all labs and assignments:

- You may not import any libraries or modules unless explicitly told to do so.
- You may not use the function "eval" unless explicitly told otherwise.
- In Racket, you may not use any iterative or mutating functionality unless explicitly allowed. Iterative functions in Racket are functions like `loop` and `for`. Mutating functions in Racket have a `!` in their names, like `set!`. If use the materials discussed in class and do not go out of your way to find these functions, your code should be okay.
- The `(provide ...)` and `module (...) where` code in your files are very important. Please follow the instructions, and don't modify those lines of code! If you do so, your code will not be able to run and you will receive a grade of zero.
- Do not change the default Haskell language settings (i.e. by writing an additional line of code in the first line of your file)

Breaking any of these above rules can result in a grade of 0.

Moreover, code that cannot be imported (e.g., due to a syntax error, compilation error, or runtime error during import) will receive a grade of zero. Please make sure to run and test all of your code before your final submission.

You are permitted (and encouraged!) to write helper functions freely.