PLC: Homework 1 [90 points]

Due date: Wednesday, September 6th, 9pm
3 extra-credit points if you turn in by Tuesday, September 5th, 9pm

About This Homework

For this homework, you will practice writing functions by recursion and pattern-matching in Haskell.

How to Turn In Your Solution

I have created a `hw1` subdirectory in every student's personal repo, and added two files there: `Hawkid.hs` (where "Hawkid" is replaced with your actual Hawkid), and `Operations.hs`. The first of these contains a datatype generated randomly from a hash of your Hawkid. So every student will be writing code based on a different datatype. The goal is to discourage inappropriate copying of code. You will modify just the `Operations.hs` file, for your solution to this homework.

As for hw0, you can check that you have submitted correctly by going to the URL for your subversion repository. Also, as for hw0, please use exactly the file names we are requesting (so do not change the names of these files).

Partners Allowed

You may work by yourself or with one partner (no more). Each partner should submit the solution, and each solution will be graded independently. Partners are, of course, allowed to submit exactly the same solution (since the partners are both working together on this single solution). You are not required, however, to submit the same solution as your partner, if you choose not to for whatever reason. Also, please create a file called `partner.txt` containing just the Hawkid of your partner, and include this with your submission (i.e., commit it using subversion). Each partner should create and submit this file.

How To Get Help

You can post questions in the `hw1` section on Piazza.

You are also welcome to come to our office hours. See the course's Google Calendar, linked from the Resources tab of the Resources page on Piazza, for the locations and times for office hours.

1 Reading

Read Chapters 3, 4, and 6 (Chapter 5 is optional) of the required book, *Programming in Haskell*, by Graham Hutton.

2 Problems

The file `Hawkid.hs` (where `Hawkid` is replaced with your actual Hawkid) conatins a datatype generated randomly from a hash of your hawkid. The datatype has a number of different constructors, taking arguments of different types. You will write code to carry out various operations with data from this datatype. Your first step is to look at that datatype. Then proceed to `Operations.hs`.

The file `Operations.hs` contains nine definitions which you must modify as follows [10 points each]:

1. `sample`: create a sample element of your datatype (the one defined in `Hawkid.hs` where `Hawkid` is your actual Hawkid). Your sample element must have at least four constructors.
2. `size`: this function should return the number of constructors in the input element it is given of your datatype.
3. `depth`: this function should return the maximum number of constructors on any path from the top of the element to a leaf node.
4. `dropN`: this function should take an element of your datatype, and return a new version of that element where all occurrences of the `N` constructor have been dropped.
5. `countB`: this function should count the number of `B` constructors in the input.
6. `ltoB`: this function should convert every subtree that looks like `L x` for some `x`, to be instead `B`. So you will replace all uses of `L` with just `B`. Notice that this enables the data to be typed with type parameter `b` instead of `a`, because only `L` takes an input of the type parameter to your datatype.
7. `toString`: this function should generate a fully parenthesized string representation of the input element, where constructors are printed out just as themselves. (Note that you must really write this function; you are not allowed to derive `Show` and then just call `show`).
8. `mapHawkid`: this takes in a function from `a` to `b` and should apply that function to the values stored under the `L` constructor.
9. `substB`: this takes in two elements `x` and `y` of your datatype, and should replace every use of `B` that you find in `y` with `x`. So you are substituting `x` for `B` in `y`.