CS 3FP3: Functional Programming

Due on Monday February 8th, 2021

Dr. Jacques Carette

Idea

The goals of this assignment are:

1. Deepen your understanding of higher-order functions (sometimes through proofs)
2. Work more with Algebraic Types

The Task

Do each of the following (but do pay attention to the submission requirements below)

1. Given

   ```
   iter :: Integer -> (a -> a) -> (a -> a)
   iter n f
     | n > 0     = f . iter (n-1) f    -- iter.1
     | otherwise = id                  -- iter.2
   ```

   then, by induction, prove that for *all natural numbers* n,

   ```
   iter n id = id
   ```

   Hint: saturation and extensionality!

2. Prove that for all ys and zs the equation

   ```
   map f (ys ++ zs) = map f ys ++ map f zs
   ```

   Hint: you only need a single induction.

3. Prove that for all finite lists xs and functions f,

   ```
   concat (map (map f) xs) = map f (concat xs)
   ```

   where

   ```
   concat :: [[a]] -> [a]
   concat = foldr (++) []   -- concat.1
   ```

   and **foldr**, ++ as defined in class. For certainty, give the full definition where you attach labels to each line.

4. Prove that for all finite lists xs that

   ```
   filter p (filter q xs) = filter (p &&& q) xs
   ```

   where

   ```
   p &&& q = \x -> p x && q x
   ```

For the next 3 questions, use the following definition of Expr:

```
data Expr =
    Lit Integer
  | Expr :+: Expr
  | Expr :-: Expr
```

5. Define the function

   ```
   size :: Expr -> Integer
   ```

   which counts the number of *operators* in an expression.

6. Add the operations of multiplication and integer division to the type Expr (call it Expr'), and define the following functions:

   ```
   show' :: Expr' -> String
   size' :: Expr' -> Integer
   eval' :: Expr' -> Integer
   ```

   by hand, where

   - **show'** give a Haskell-readable rendering of an Expr' (i.e. useful for cut-and-paste)
   - size' is as above
   - eval' *evaluates* the expression.

   Answer the following question in comments: what does your function do when asked to perform a division by zero?

7. Instead of adding extra constructors to the Expr type, it is possible to factor the definitions as

   ```
   data Expr2 =
       Lit2 Integer
     | Op Ops Expr2 Expr2

   data Ops = Add | Sub | Mul | Div
   ```

   Implement **show**, eval and size for this type. Discuss the changes you have to make to your definitions if you add the extra operation Mod for *remainder on integer division*.

8. Generalize **either** (from standard library) to **join** with type

   ```
   join :: (a -> c) -> (b -> d) -> Either a b -> Either c d
   ```

   Note: there is an elegant solution that uses **either** to define **join**, but this is not the only valid solution.

For the next question, we need a generalized tree, this time defined as

```
data GTree a = Leaf a | Gnode [GTree a]
```

8. Define functions

   (a) to count the number of leaves in a `GTree`;

   (b) to find the depth of a `GTree`;

   (c) to sum a numeric `GTree Int`;

   (d) to find whether an element appears in a `GTree`;

   (e) to map a function over the elements at the leaves of a `GTree`; and

   (f) to flatten a `GTree` to a list.

   In each case give the type of the function that you have defined.

9. **Bonus:** Add to the regular expressions the facility to name substrings that match particular sub-expressions, so that instead of returning a **Bool**, a RegExp (as defined in class) will return a *list* of bindings of names to substrings.

   Why a list? First, it allows for no matching to happen (via the empty list), or for *multiple* matches, which can also happen as matching the regular expressions e1 <**> e2 and star r can succeed in multiple different ways.

Submission Requirements

- Must contain either a text file `A2_macid.lhs` (with your id, i.e. your email address, I am 'carette', substituted in) with all your Haskell code, or an org file `A2_macid.org` also with everything.
- The names of the file **does** matter.
- Code which **does not compile** is worth **0** marks.
- if you have looked things up online (or in a book) to help, document it in your code. If you have asked a friend for help, document that too.

Notes

For this assignment, you may use anything from the standard library.

More Bonus

You should do the assignment itself, then the following on top of that.

1. Do the assignment using Org mode instead of using `.lhs`. Please hand in a tangled `.lhs` file as well as the `.org` file.
2. Do the assignment in Agda (or Idris). Including the properties – but these you should *prove*. Worth up to 10 final marks. Hand in either `.lagda` or `.lidr` files. Or both if you are super-keen.