Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
http://courses.softlab.ntua.gr/pl2/

Γλώσσες Προγραμματισμού II

Αν δεν αναφέρεται διαφορετικά, οι ασκήσεις πρέπει να παραδίδονται στους διδάσκοντες σε ηλεκτρονική μορφή μέσω του συνεργατικού συστήματος ηλεκτρονικής μάθησης `moodle.softlab.ntua.gr`. Η προθεσμία παράδοσης θα τηρείται αυστηρά. Έχετε δικαίωμα να καθυστερήσετε το πολύ μία άσκηση.

Άσκηση 5 Τα δέντρα και πώς να τα διπλώνετε, με μονάδες και χωρίς

Προθεσμία παράδοσης: 22/3/2015

Έστω στη γλώσσα Haskell ο τύπος `Tree` του δέντρου με οσαδήποτε παιδιά ανά κόμβο, ο ίδιος που εξετάσαμε στο παράδειγμα με τη διάσχιση DFS/BFS, στην παράδοση της 10/12/2014.

```
data Tree a = T a [Tree a]
```

1. Υλοποιήστε την αναδρομική συνάρτηση `foldTree` που "διπλώνει" ένα δέντρο με τον ίδιο τρόπο όπως οι `foldl` και `foldr` διπλώνουν λίστες.

   ```
   foldTree :: (a → [b] → b) → Tree a → b
   ```

   Η πρώτη παράμετρος της `foldTree` είναι μια συνάρτηση `f` τέτοια ώστε `f x bs` είναι το επιθυμητό αποτέλεσμα για ένα δέντρο που περιέχει την τιμή `x` στη ρίζα του και που τα επιθυμητά αποτελέσματα για τα παιδιά του δίνονται στη λίστα `bs`.

2. Στη συνέχεια, χρησιμοποιώντας τη `foldTree`, υλοποιήστε τις παρακάτω συναρτήσεις:

   - `sizeTree t`: το πλήθος των κόμβων του δέντρου `t`.
   - `heightTree t`: το "ύψος" του δέντρου `t`, δηλαδή το μήκος του μεγαλύτερου μονοπατιού από τη ρίζα σε κάποιο φύλλο.
   - `sumTree t`: το άθροισμα των τιμών των κόμβων του δέντρου `t`.
   - `maxTree t`: η μέγιστη των τιμών των κόμβων του δέντρου `t`.
   - `inTree x t`: ελέγχει αν η τιμή `x` βρίσκεται σε κάποιο κόμβο του δέντρου `t`.
   - `nodes t`: μια λίστα που περιέχει τις τιμές όλων των κόμβων του δέντρου `t`.
   - `countTree f t`: το πλήθος των κόμβων του δέντρου `t`, οι τιμές των οποίων ικανοποιούν το κατηγόρημα `f`.
   - `leaves t`: μια λίστα που περιέχει τις τιμές όλων των φύλλων του δέντρου `t`.
   - `mapTree f t`: το δέντρο που προκύπτει αν σε κάθε κόμβο του δέντρου `t` αντικαταστήσουμε την τιμή `x` με `f x`.

   Η υλοποίηση των παραπάνω συναρτήσεων πρέπει να γίνει χωρίς τη χρήση αναδρομής με μία και μόνο κλήση στη συνάρτηση `foldTree`, με κατάλληλη παράμετρο. (Για τις περισσότερες από αυτές, η υλοποίηση θα είναι της μίας γραμμής.) Οι τύποι τους πρέπει να είναι οι εξής:

```
sizeTree   :: Num b ⇒ Tree a → b
heightTree :: (Ord b, Num b) ⇒ Tree a → b
sumTree    :: Num a ⇒ Tree a → a
maxTree    :: (Ord a) ⇒ Tree a → a
inTree     :: Eq a ⇒ a → Tree a → Bool
nodes      :: Tree a → [a]
countTree  :: (a → Bool) → Tree a → Integer
leaves     :: Tree a → [a]
mapTree    :: (a → b) → Tree a → Tree b
```

3. Στην παράδοση της 10/12/2014 είδαμε τις συναρτήσεις `dfn` και `bfn`, που "διακοσμούν" ένα δέντρο, προσθέτοντας σε κάθε κόμβο τη σειρά με την οποία τον επισκέπεται η διάσχιση DFS και BFS, αντίστοιχα. Στην παράδοση της 17/12/2014 είδαμε πώς η πρώτη από αυτές μπορεί να γραφεί απλούστερα χρησιμοποιώντας το state monad. Ανατρέξτε στη σελίδα του μαθήματος για τον κώδικα αυτών των παραδειγμάτων.

   Η ίδια τεχνική, με τη χρήση ενός κατάλληλου monad, μπορεί να γίνει και σε συνδυασμό με τη συνάρτηση `foldTree`, αν σε αυτήν ο τύπος `b` επιλεγεί να είναι κατάλληλος τύπος που προέρχεται από αυτό το monad.

   Υλοποιήστε ξανά τις συναρτήσεις `dfn` και `bfn`, βασισμένοι στις υλοποιήσεις της παράδοσης της 17/12/2014 στη σελίδα του μαθήματος, χρησιμοποιώντας το `Control.Monad.State` και κάνοντας μία κλήση στη `foldTree`, με κατάλληλη (μη αναδρομική) παράμετρο. Οι τύποι των δύο συναρτήσεων πρέπει να είναι:

   ```
   dfn :: Tree a → Tree (a, Integer)
   bfn :: Tree a → Tree (a, Integer)
   ```

4. Με παρόμοιο τρόπο, υλοποιήστε δύο ακόμη συναρτήσεις, χρησιμοποιώντας το `Control.Monad.Reader` και μία κλήση στη `foldTree`, με κατάλληλη (μη αναδρομική) παράμετρο:

   - `trimTree n t`: το αποτέλεσμα του "κλαδέματος" του δέντρου `t` στο ύψος `n`, δηλαδή το δέντρο `t` αν αφαιρέσουμε όλους τους κόμβους που η απόστασή τους από τη ρίζα υπερβαίνει το `n`.
   - `path l t`: η τιμή του κόμβου στον οποίο φθάνουμε αν ξεκινήσουμε από τη ρίζα του δέντρου `t` και ακολουθήσουμε το μονοπάτι που ορίζεται από τη λίστα `l`. Για παράδειγμα, η λίστα `[0,1]` μας οδηγεί στο δεύτερο παιδί (1) του πρώτου παιδιού (0) της ρίζας. Θεωρήστε δεδομένο ότι θα είναι δυνατό να ακολουθήσετε το μονοπάτι `l` μέσα στο δέντρο `t`, διαφορετικά η συνάρτησή σας μπορεί να προκαλεί σφάλμα εκτέλεσης.

   Ο τύπος των δύο συναρτήσεων πρέπει να είναι:

   ```
   trimTree :: Integer → Tree a → Tree a
   path     :: [Int] → Tree a → a
   ```

Υπόδειξη. Ίσως βρείτε χρήσιμη τη συνάρτηση `sequence` της βιβλιοθήκης της Haskell.