Γλώσσες Προγραμματισμού ΙΙ

Άσκηση 4 (Έλεγχος διάσχισης δέντρων)

Δέντρο σε Haskell:

```
data Tree a = Node a [Tree a]
```

1.

Συνάρτηση `arbitrary`:

```
arbitrary = sized arbWithSize
```

Όπου η βοηθητική συνάρτηση `arbWithSize` ορίζεται ως εξής:

```
arbWithSize 0 = do
                x <- arbitrary
                return (Node x [])
arbWithSize n = frequency [ (1, do
                                 x <- arbitrary
                                 return (Node x []))
                          , (9, do
                                 x <- arbitrary
                                 (Positive m) <- arbitrary
                                 let k = n `div` m
                                 ts <- replicateM m (arbWithSize k)
                                 return (Node x ts))
                            ]
```

Η ιδέα για την γενική περίπτωση είναι η εξής:

- Διαλέγουμε έναν τυχαίο αριθμό x (θεωρούμε δέντρα τύπου `Tree Int`) ως τιμή για την ρίζα.
- Διαλέγουμε έναν τυχαίο **θετικό** αριθμό m που είναι το πλήθος των παιδιών τής ρίζας.
- Αναδρομικά κατασκευάζουμε m υπο-δέντρα μεγέθους n `div` m.

Σχόλιο:

Επιλέγουμε με 1/10 πιθανότητα την κατασκευή φύλλου ενώ με 9/10 πιθανότητα την κατασκευή υπο-δέντρων. Έτσι, τα δέντρα που προκύπτουν δεν είναι πάντα balanced.

2.

Συνάρτηση `shrink = shrinkTree`

```
shrinkTree (Node _ []) = [] -- do nothing
shrinkTree (Node x ts) =
    [(Node x [])] ++  -- reduce to a leaf
    ts ++             -- reduce to a child
    [(Node x ts') | ts' <- mapM shrinkTree ts] -- shrink children
```

Επεξήγηση:

Όταν βρούμε ένα test στο οποίο μια ιδιότητα δεν επαληθεύεται θα δοκιμάσουμε τα εξής shrinks.

- Αντικατάσταση τού δέντρου (Node x ts) με το φύλλο (Node x []).
- Αντικατάσταση τού δέντρου με ένα από τα παιδιά τής ρίζας.
- Αναδρομικά εφαρμογή shrink στα παιδιά.

3.

Για τον έλεγχο των συναρτήσεων bfs, dfs χρησιμοποιήσαμε τις ακόλουθες ιδιότητες:

- prop_tree_height
- prop_tree_size
- prop_tree_root

Η λειτουργία τους είναι προφανής.

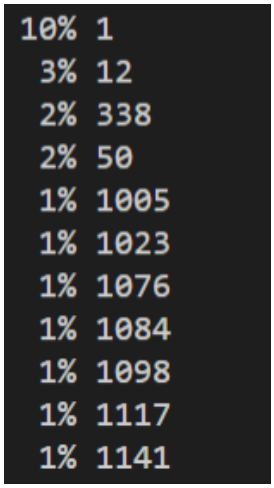
4.

Εκτός από τις ζητούμενες συναρτήσεις υλοποιήσαμε και την συνάρτηση `dummy` η οποία κάνει annotate όλους τους κόμβους με τον αριθμό 42. Η συνάρτηση αυτή ικανοποιεί τις ιδιότητες `prop_tree_height` και `prop_tree_size` αλλά όχι την `prop_tree_root`. Τρέχοντας το πρόγραμμα παίρνουμε τα εξής αποτελέσματα:

```
"-- testing bfs function --"
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
"-- testing dfs function --"
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
"-- testing dummy function --"
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
*** Failed! Falsified (after 1 test):
Node 0 []
```

Όπως αναμέναμε οι συναρτήσεις bfs, dfs περνάνε όλα τα test ενώ η dummy δεν περνάει το τελευταίο που ελέχγει την ιδιότητα `prop_tree_root`.

Κάνοντας collect τα size των δέντρων που παρήγαγε το quickCkeck παρατηρούμε ότι κατασκευάζουμε τόσο μεγάλα όσο και μικρά δέντρα. Ενδεικτικά έχουμε:



(Σχόλιο: Ίσως θα έπρεπε να αλλάξουμε τα frequency (αυξάνοντας το 9) αφού το 10% των δέντρων που παρήχθησαν είχαν size 1...)

5. (Υλοποίηση τής συνάρτησης merge)

6.

Για τον έλεγχο των συναρτήσεων merge και wrong χρησιμοποιούμε την ιδιότητα prop_tree_merge με την οποία ελέγχουμε ότι η ρίζα τού merged tree έχει πλήθος παιδιών ίσο με το μέγιστο πλήθος παιδιών των ριζών των δύο δέντρων που γίνονται merge.