Συναρτησιακός Προγραμματισμός 2008
Δεύτερο Φύλλο Ασκήσεων

Οι ασκήσεις αυτές είναι ατομικές
Οι ασκήσεις αντιστοιχούν στο 15% του βαθμού στο μάθημα.
Συνολικό Άθροισμα Βαθμών: 150
Προθεσμία Παράδοσης: Τρίτη 20 Μαΐου

1. **Βαθμοί: 25**. Στις Σημ. 4, είδαμε τη δημιουργία της κλάσης `Condition` που μας επιτρέπει να χρησιμοποιούμε αριθμούς, λίστες και ζεύγη ως αληθοτιμές στη συνάρτηση `cond` που αποτελεί γενίκευση της `if then else`.

   Θα θέλαμε να επεκτείνουμε την κλάση αυτή, έτσι ώστε να μπορούμε να χειριστούμε μία λίστα αληθοτιμών στην `cond` είτε ως σύζευξη είτε ως διάζευξη των στοιχείων της. Για παράδειγμα, η λίστα `[2::Int,False]` θα πρέπει να λειτουργεί σαν `True` στην περίπτωση που τη βλέπουμε σα διάζευξη (το `2::Int` λειτουργεί σαν `True`), ενώ θα πρέπει να λειτουργεί σα `False` στην περίπτωση που τη βλέπουμε σα σύζευξη.

   Το πρόβλημα είναι ότι οι λίστες από μόνες τους δε μπορούν να μας πουν αν τις βλέπουμε σα σύζευξη ή σα διάζευξη. Επιπλέον, οι λίστες έχουν ήδη ένα χειρισμό στην `Condition`: μία λίστα συμπεριφέρεται σαν `True` αν και μόνον αν δεν είναι κενή.

   Χρησιμοποιώντας αλγεβρικούς τύπους, λύστε αυτό το πρόβλημα και επεκτείνετε την κλάση `Condition` ώστε να διατηρήσει τη λειτουργικότητα που έχει τώρα, αλλά να υποστηρίζει και σύζευξη / διάζευξη στοιχείων μίας λίστας.

   Θα βρείτε τον κώδικα ορισμού της κλάσης `Condition` όπως είναι τώρα στο αρχείο
   `http://cgi.di.uoa.gr/~kassios/courses/fp/exercises/Condition.hs`

2. **Βαθμοί: 35**. Εμπλουτίστε τον αλγεβρικό τύπο `Expr` των Σημ. 5, Ενότ. 8.2 με μία δομή `let` που εκχωρεί τιμές σε μεταβλητές. Μετά κατασκευάστε πλήρη διερμηνέα `eval` για τις εκφράσεις αυτές. Ο διερμηνέας θα πρέπει να επιστρέφει τιμή τύπου `Maybe Int`, καθώς θα μπορούσαν να συμβούν λάθη κατά την αποτίμηση μίας έκφρασης, όπως διαίρεση με το μηδέν ή αποτίμηση μίας μεταβλητής στην οποία δεν έχει εκχωρηθεί τιμή.

3. **Βαθμοί: 50**. Ένα *πολυσύνολο (multiset)* είναι μία συλλογή δεδομένων στην οποία η σειρά δεν έχει σημασία, αλλά ο πληθυσμός έχει σημασία. Δηλαδή, ένα πολυσύνολο που περιέχει τα `1,2` είναι ίσο με ένα πολυσύνολο που περιέχει τα `2,1`, αλλά όχι ίσο με αυτό που περιέχει τα `1,2,1`. Υλοποιήστε έναν πολυμορφικό ΑΤΔ `MultiSet` για πολυσύνολα. Υποστηρίξτε την κατασκευή τουλάχιστον όλων των πεπερασμένων πολυσυνόλων και την εξαγωγή όλων των χρήσιμων πληροφοριών από ένα πολυσύνολο. Υποστηρίξτε πράξεις `union` (ένωση) και `difference` (διαφορά) φροντίζοντας να είναι άμεσες γενικεύσεις των αντίστοιχων πράξεων για σύνολα. Μπορείτε να φτιάξετε και ό,τι άλλη συνάρτηση νομίζετε ότι μπορεί να διευκολύνει τη χρήση πολυσυνόλων. Τέλος, δηλώστε τον τύπο σας ως στιγμιότυπο της κλάσης `Eq`.

4. **Βαθμοί: 10**. Χρησιμοποιήστε το τμήμα `MultiSet` που φτιάξατε για να ορίσετε μία συνάρτηση `perm` που να παίρνει δύο λίστες και να επιστρέφει `True` αν και μόνον αν η μία είναι αναμετάθεση της άλλης.

5. **Βαθμοί: 30**. Υλοποιήστε το τμήμα `ResettableVariable` με υπογραφή εξόδου:

   - `ResVar a` (Πολυμορφικός ΑΤΔ)
   - `newRV :: ResVar`
   - `value :: ResVar a -> a`
   - `set :: ResVar a -> a -> ResVar`
   - `reset :: ResVar a -> ResVar`

   και με συμβόλαιο (για κάθε `r::ResVar a` και `v::a`)

   ```
      value(set r v) = v
   && value(reset(set r v)) = value r
   && reset . reset = reset
   ```

   Αποδείξτε ότι το συμβόλαιο ικανοποιείται.