Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Επώνυμο:
Όνομα:
Αριθμός Μητρώου:

1	1	
2	2	
3	1	
4	1	
5	1	
6	2	
Σ	8	

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι *Εαρινό 2017*

Τελικό Διαγώνισμα (Επαναληπτική Εξέταση)

Το διαγώνισμα αυτό έχει ερωτήσεις 8 βαθμών συνολικά.

1. Γραμματικές (0.2 + 0.2 + 0.6 = 1 βαθμός)

Έστω η παρακάτω γραμματική χωρίς συμφραζόμενα η οποία γεννά ένα υποσύνολο των εκφράσεων μίας συναρτησιακής γλώσσας προγραμματισμού σαν την ML.

```
<Expr>  ::= fn <Var> => <Expr>          — ανώνυμη συνάρτηση
          | <Expr> <Expr>               — εφαρμογή συνάρτησης
          | <Var>  |  ( <Expr> )        — παράμετροι και παρενθέσεις
<Var>   ::= x | y | z
```

α) Δώστε ένα συντακτικό δένδρο για τη συμβολοσειρά: `(fn x => x) y`

β) Δείξτε ότι η γραμματική αυτή είναι διφορούμενη.

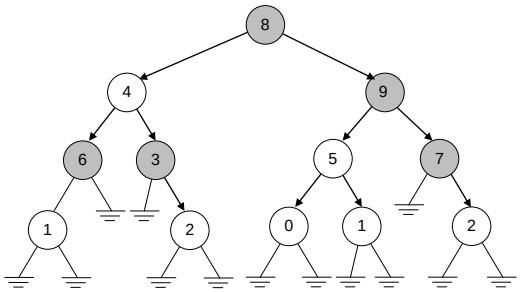
γ) Τροποποιήστε τη γραμματική ώστε οι τελεστές να έχουν τη σωστή προτεραιότητα και προσεταιριστικότητα. Η εφαρμογή συνάρτησης πρέπει να είναι αριστερά προσεταιριστική. Το σώμα μίας ανώνυμης συνάρτησης πρέπει να εκτείνεται προς τα δεξιά όσο περισσότερο γίνεται. Η χρήση ανώνυμων συναρτήσεων σε εφαρμογές πρέπει να γίνεται μέσα σε παρενθέσεις. Π.χ., η έκφραση `z (fn x => x y z)` πρέπει να θεωρείται ισοδύναμη με την `z (fn x => ((x y) z))`.

2. Προγραμματισμός σε ML (1 + 1 = 2 βαθμοί)

α) Έστω ο παρακάτω τύπος του πολυμορφικού δυαδικού δένδρου **`'a tree`** που περιέχει ως πληροφορία στοιχεία οποιουδήποτε τύπου **`'a`**.

```
datatype 'a tree = node of 'a * 'a tree * 'a tree | empty
```

Έστω ένα δένδρο αυτού του τύπου, το οποίο μας ενδιαφέρει να είναι ισοζυγισμένο με την εξής έννοια: κάθε κόμβος (node) θέλουμε να έχει ίσο πλήθος κόμβων στο αριστερό του υποδένδρο και στο δεξιό του υποδένδρο. Ένας κόμβος του δένδρου που δεν έχει αυτή την ιδιότητα ονομάζεται «εκτός ισορροπίας».



Να γράψετε σε ML μία *κομψή και αποδοτική* συνάρτηση **`countUnbalanced`** η οποία να δέχεται ως παράμετρο ένα τέτοιο δένδρο και να επιστρέφει πόσοι κόμβοι του είναι εκτός ισορροπίας.

Παράδειγμα: Για το δένδρο του διπλανού σχήματος η συνάρτησή σας πρέπει να επιστρέφει 5.
Εξήγηση: Οι κόμβοι 8, 6, 3, 9 και 7, που είναι χρωματισμένοι με γκρι, είναι εκτός ισορροπίας. Π.χ., ο κόμβος 9 είναι εκτός ισορροπίας γιατί το αριστερό του υποδένδρο (με ρίζα τον κόμβο 5) έχει 3 κόμβους ενώ το δεξιό του υποδένδρο (με ρίζα τον κόμβο 7) έχει 2.

β) Στην εξέταση κάποιου μαθήματος σχετικού με συναρτησιακό προγραμματισμό ζητήθηκε να γραφεί σε ML μία *κομψή και αποδοτική* (ε-χεμ) συνάρτηση **`alternatingEvenOdd`** η οποία να δέχεται ως όρισμα μια λίστα από ακέραιους αριθμούς και να επιστρέφει **`true`** αν και μόνο αν η λίστα αυτή αποτελείται από άρτιους και περιττούς αριθμούς εναλλάξ, ξεκινώντας με άρτιο, διαφορετικά **`false`**. Ένας σπουδαστής απάντησε γράφοντας τα εξής:

```
val alternatingEvenOdd =
  let fun aux p [] = true
        | aux p (h :: t) = p h andalso aux (not o p) t
  in  aux (fn x => x mod 2 = 0)
  end
```

βραχυκύκλωση

άρνηση $bool \to bool$

σύνθεση συναρτήσεων

β1) Είναι σωστή η λύση του σπουδαστή, δηλαδή επιστρέφει την επιθυμητή τιμή για κάθε τιμή της παραμέτρου; Αιτιολογήστε σύντομα την απάντησή σας. Μπορείτε να χρησιμοποιήσετε έναν κατάλληλο επαρκή ισχυρισμό ή αντιπαράδειγμα.

β2) Είναι αποδοτική (σε χρόνο ή/και μνήμη) η λύση του σπουδαστή για μεγάλες σε μήκος λίστες; Αιτιολογήστε σύντομα την απάντησή σας.

3. Συμπερασμός τύπων στην ML (4 * 0.25 = 1 βαθμός)

Συμπληρώστε τους τύπους των παρακάτω συναρτήσεων:

```
fun foo x y   = x (y + 1) y
fun bar x y   = y (bar y x)
fun doh x y z = y (z x) + 1
fun ugh x y z = z x (y x)
```

	Τύπος
foo	
bar	
doh	
ugh	

4. Πέρασμα παραμέτρων (4 * 0.25 = 1 βαθμός)

Έστω το παρακάτω πρόγραμμα σε μια υποθετική γλώσσα που μοιάζει με τη C. Προσέξτε ότι η αρίθμηση των στοιχείων των πινάκων ξεκινάει από το μηδέν.

```
int A[3] = {0, 2, 1};

void f(int x, int y) {
  x++; A[1]--; y++;
  print(x, y, A[0], A[1], A[2]);
}

void main() {
  f(A[0], A[A[1]]);
  print(A[0], A[1], A[2]);
}
```

Συμπληρώστε τον πίνακα με τις οκτώ τιμές που θα εκτυπώσει το πρόγραμμα για καθεμία από τις παρακάτω τεχνικές περάσματος παραμέτρων της συνάρτησης **f**.

κλήση με τιμή (by value)								
κλήση κατ' αναφορά (by reference)								
κλήση με τιμή-αποτέλεσμα (by value-result)								
κλήση κατ' όνομα (by name)								

5. Ερωτήσεις πολλαπλής επιλογής (4 * 0.25 = 1 βαθμός)

Κάθε λάθος απάντηση αφαιρεί 0.1 βαθμό από αυτό το θέμα.

α) Έστω το διπλανό πρόγραμμα σε μια υποθετική γλώσσα που μοιάζει με τη C.

α1) Αν η γλώσσα υλοποιεί στατικές εμβέλειες, το πρόγραμμα θα εκτυπώσει:

A. 1, 1 B. 1, 2 Γ. 2, 1 Δ. 2, 2

α2) Αν η γλώσσα υλοποιεί δυναμικές εμβέλειες, το πρόγραμμα θα εκτυπώσει:

A. 1, 1 B. 1, 2 Γ. 2, 1 Δ. 2, 2

```
int a = 1;

int f() { print(a); }

int g() {
  int a = 2;
  f();
}

void main() {
  g();
  f();
}
```

β) Έστω το διπλανό πρόγραμμα σε μια υποθετική γλώσσα που μοιάζει με τη Java.

β1) Αν η γλώσσα υλοποιεί στατική αποστολή μεθόδων (static dispatch), το πρόγραμμα θα εκτυπώσει:

A. 42, 42 B. 42, 17 Γ. 17, 42 Δ. 17, 17

β2) Αν η γλώσσα υλοποιεί δυναμική αποστολή μεθόδων (dynamic dispatch), το πρόγραμμα θα εκτυπώσει:

A. 42, 42 B. 42, 17 Γ. 17, 42 Δ. 17, 17

```
class B {
  int x;
  B() { x = 42; }
  void f() { print(x); }
};

class D extends B {
  int x;
  D() { x = 17; }
  void f() { print(x); }
}

void main() {
  B b = new B(); b.f();
  B d = new D(); d.f();
}
```

6. Προγραμματισμός σε Prolog (1 + 1 = 2 βαθμοί)

α) Στην εξέταση του Ιουνίου ζητήθηκε από τους σπουδαστές να γράψουν σε Prolog τον ορισμό του κατηγορήματος **`running_sum(L, S).`** Το κατηγόρημα αυτό πρέπει να επιτυγχάνει όταν τα L και S είναι λίστες ακέραιων αριθμών και η λίστα S περιέχει τα τρέχοντα αθροίσματα των όρων της λίστας L. Δηλαδή, το N-οστό στοιχείο της λίστας S θα πρέπει να είναι ίσο με το άθροισμα των N πρώτων στοιχείων της λίστας L. Παράδειγμα χρήσης:

```
?- running_sum([5, 3, 7, -2, 0, 4], S).
S = [5, 8, 15, 13, 13, 17].
```

Ένας σπουδαστής έδωσε την εξής απάντηση:

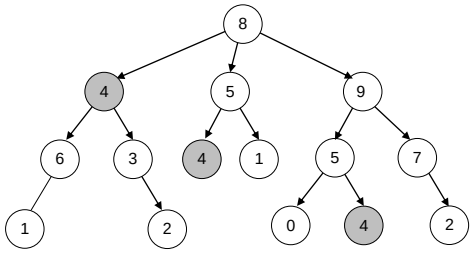
```
running_sum(L, S) :- helper(L, S, 0).

helper([], [], _).
helper([H1|T1], [H2|T2], A) :-
  H2 is H1 + A,
  helper(T1, T2, +(H1, A)).
```

α1) Είναι σωστή η λύση του σπουδαστή, δηλαδή έχει την επιθυμητή συμπεριφορά, όταν ως πρώτο όρισμα δίδεται μία λίστα ακεραίων αριθμών; Αιτιολογήστε σύντομα την απάντησή σας. Μπορείτε να χρησιμοποιήσετε ένα κατάλληλο επαρκή ισχυρισμό ή αντιπαράδειγμα.

α2) Είναι αποδοτική (σε χρόνο ή/και μνήμη) η λύση του σπουδαστή για μεγάλες σε μήκος λίστες; Αιτιολογήστε σύντομα την απάντησή σας.

β) Έστω ότι δένδρα (όχι κατ' ανάγκη δυαδικά) σαν αυτό του διπλανού σχήματος αναπαρίστανται στην Prolog ως εξής. Κάθε κόμβος του δένδρου είναι μία λίστα που ως πρώτο στοιχείο έχει την πληροφορία που περιέχεται στον κόμβο (στο διπλανό σχήμα είναι ακέραιοι αριθμοί) και ως επόμενα στοιχεία έχει τα παιδιά του κόμβου (αν υπάρχουν).



Για παράδειγμα, το δένδρο του διπλανού σχήματος αναπαρίσταται από τον όρο:

```
T = [8, [4, [6, [1]], [3, [2]]], [5, [4], [1]],
        [9, [5, [0], [4]], [7, [2]]]]
```

Γράψτε σε Prolog τον ορισμό του κατηγορήματος **`find_depth(T, X, D)`**. Το πρώτο όρισμα θα είναι ένα δένδρο, το δεύτερο θα είναι μία πληροφορία προς αναζήτηση και το τρίο θα είναι ένας ακέραιος αριθμός. Το κατηγόρημα πρέπει να επιτυγχάνει όταν κάποιος κόμβος του δένδρου T που βρίσκεται σε βάθος D περιέχει την πληροφορία X. Θεωρούμε ότι η ρίζα βρίσκεται σε βάθος 1.

Ακολουθούν δύο παραδείγματα χρήσης, όπου θεωρούμε ότι η μεταβλητή T έχει ενοποιηθεί με τον παραπάνω όρο που αντιστοιχεί στο δένδρο του σχήματος. Στο πρώτο (αριστερά) αναζητάται η πληροφορία 4 στο δένδρο T (υπάρχει τρεις φορές, σε βάθη 2, 3 και 4 — βλ. γκρι κόμβους).

```
?- find_depth(T, 4, D).
D = 2 ;
D = 3 ;
D = 4 ;
false.
```

```
?- find_depth(T, X, 2).
X = 4 ;
X = 5 ;
X = 9 ;
false.
```

Καλή επιτυχία!