ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εαρινό 2016

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Τελικό Διαγώνισμα (Κανονική Εξέταση)

Το διαγώνισμα αυτό έχει ερωτήσεις 8 βαθμών συνολικά.

1. Γραμματικές (0.1 + 0.2 + 0.25 + 0.2 = 0.75 βαθμοί)

Έστω η παρακάτω γραμματική χωρίς συμφραζόμενα η οποία γεννά όλες τις κανονικές εκφράσεις. Προσέξτε ότι η γραμματική έχει *πέντε* κανόνες για το <RE>. Στον πρώτο κανόνα της το σύμβολο | είναι τερματικό και συμβολίζει την διάζευξη (alternation ή union) δύο κανονικών εκφράσεων

```
<RE> ::= <RE> | <RE>
       |  <RE> • <RE>
       |  <RE> *
       |  ( <RE> )
       |  <letter>
<letter> ::= a | b | ... | z
```

α) Δώστε ένα συντακτικό δένδρο για τη συμβολοσειρά `(a•b|b)*`

β) Δείξτε ότι η γραμματική αυτή είναι διφορούμενη.

γ) Τροποποιήστε τη γραμματική έτσι ώστε οι τελεστές να έχουν τη σωστή προτεραιότητα (αυτή που περιμένουμε) για τις πράξεις που συμβολίζουν.

δ) Τι προσεταιριστικότητα δίνει η απάντησή σας στο προηγούμενο υποερώτημα στους τελεστές της διάζευξης και της παράθεσης; Δικαιολογήστε σύντομα την απάντησή σας.

2. Προγραμματισμός σε ML (0.75 + 1 = 1.75 βαθμοί)

α) Να γράψετε σε ML τη συνάρτηση `listify` η οποία δέχεται ως όρισμα μια λίστα από ακεραίους και επιστρέφει μια λίστα από λίστες από ακεραίους τέτοιες ώστε όλες να έχουν τα στοιχεία τους σε μη φθίνουσα σειρά και η συνένωση όλων των λιστών να μας δίνει τη λίστα εισόδου. Η συμπεριφορά της συνάρτησης γίνεται πιο εύκολα κατανοητή από τα παρακάτω παραδείγματα:

```
- listify [1,2,4,3,6,5,7];
val it = [[1,2,4],[3,6],[5,7]] : int list list

- listify [3,5,1,8,9,2,1,0,1];
val it = [[3,5],[1,8,9],[2],[1],[0,1]] : int list list
```

β) Να γράψετε σε ML τη συνάρτηση `lenfreqsort` η οποία δέχεται ως όρισμα μια λίστα από λίστες και επιστρέφει ως αποτέλεσμα τη λίστα εισόδου ταξινομημένη ως προς τη συχνότητα εμφάνισης του μήκους των λιστών, όπου οι λίστες με μήκος το οποίο εμφανίζεται σπάνια είναι πριν από τις λίστες των οποίων το μήκος εμφανίζεται πιο συχνά. Τρία παραδείγματα παρακάτω:

```
- lenfreqsort [[1,2,4],[3,6],[5,7]];
val it = [[1,2,4],[3,6],[5,7]] : int list list

- lenfreqsort [[3,5],[1,8,9],[2],[1],[0,1]];
val it = [[1,8,9],[3,5],[0,1],[2],[1]] : int list list
```

```
- lenfreqsort [[1,2,3],[4,5],[6,7,8],[1,2],[0],[4,2]];
val it = [[0],[1,2,3],[6,7,8],[4,5],[1,2],[4,2]] : int list list
```

Αναλύουμε το τελευταίο από τα τρία παραδείγματα. Υπάρχει μόνο μία λίστα με μήκος ένα η οποία εμφανίζεται στην αρχή της λίστας εξόδου. Μετά εμφανίζονται οι δύο λίστες μήκους τρία και τέλος οι τρεις λίστες με μήκος δύο των οποίων το μήκος είναι το πιο συχνά εμφανιζόμενο. Προσέξτε ότι η τιμή επιστροφής δεν είναι μονοσήμαντα ορισμένη. Η συνάρτησή σας θα μπορούσε κάλλιστα να επιστρέφει π.χ. `[[0],[6,7,8],[1,2,3],[4,2],[1,2],[4,5]]` ως αποτέλεσμα στο τελευταίο παράδειγμα.

Για το ερώτημα αυτό μπορείτε να θεωρήσετε ως δεδομένες τη γνωστή συνάρτηση **`length`**, τις συναρτήσεις **`map`** και **`fold`** αν τις χρειαστείτε, και μια συνάρτηση **`sort fcmp lst`** η οποία δέχεται ως ορίσματα μια συνάρτηση **`fcmp`** σύγκρισης δύο στοιχείων και μια λίστα **`lst`** και επιστρέφει ως αποτέλεσμα τη λίστα από τα στοιχεία της λίστας **`lst`** ταξινομημένα με βάση τη συνάρτηση σύγκρισης **`fcmp`**, την οποία φυσικά θα πρέπει να ορίσετε κατάλληλα εσείς.

3. Συμπερασμός τύπων στην ML (3 * 0.25 = 0.75 βαθμοί)

Συμπληρώστε τον πίνακα της σελίδας απαντήσεων με τους τύπους των παρακάτω συναρτήσεων:

```
fun foo x y z = x(y(z+42))
fun bar x y   = x y (y+42)
fun baz x y   = x(y(x))
```

4. Εγγραφές δραστηριοποίησης (0.75 + 0.25 = 1 βαθμός)

Η εντολή **`f := a`** στη γραμμή 11 του παρακάτω προγράμματος καλεί τη συνάρτηση **`a`**, η οποία περνάει τη συνάρτηση **`addm`** πίσω ως αποτέλεσμα.

```
1   program ret(input, output);
2   var f: function(integer): integer;

3   function a: function(integer): integer;
4     var m: integer;
5     function addm(n: integer): integer;
6       begin  return m + n  end;
7     begin  m := 0; return addm  end

8   procedure b(g: function(integer): integer);
9     begin  writeln(g(2))  end;

10  begin
11    f := a; b(f)
12  end
```

α) Υποθέστε ότι η παραπάνω γλώσσα χρησιμοποιεί στατική εμβέλεια για όλα τα μη τοπικά ονόματα. Εξηγήστε συνοπτικά για ποιο λόγο το πρόγραμμα θα αποτύχει αν χρησιμοποιεί μόνο τη στοίβα ως χώρο αποθήκευσης.

β) Ποια είναι η έξοδος του παραπάνω προγράμματος αν χρησιμοποιηθεί και ο σωρός;

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

Το ακόλουθο πρόγραμμα χρησιμοποιεί Java arrays. (Υπενθυμίζεται ότι οι δείκτες των arrays της Java αρχίζουν από το 0).

```
int[] A = new int[2];
A[0] = 0;
A[1] = 2;
f(A[0], A[A[0]]);
```

Η συνάρτηση `f` ορίζεται ως εξής:

```
void f(int x, int y) {
  x = 1;
  y = 3;
}
```

Συμπληρώστε τον πίνακα της σελίδας απαντήσεων με τις τελικές τιμές του array A μετά την κλήση και τη λήξη της `f` για καθεμία από τις παρακάτω τεχνικές περάσματος παραμέτρων. (Πιθανώς να υπάρχουν περισσότερες από μία ορθές απαντήσεις – υποθέστε λοιπόν ότι το πέρασμα των παραμέτρων γίνεται από αριστερά προς τα δεξιά.)

α) κλήση κατ' αναφορά
β) κλήση τιμή-αποτέλεσμα
γ) κλήση με επέκταση μακροεντολών
δ) κλήση κατ' όνομα

6. Σχεδιασμός γλωσσών προγραμματισμού (1 βαθμός)

Για κάποια γλώσσα προγραμματισμού που ξέρετε εκτός των ML, Java και Prolog, αναφέρετε τη γλώσσα και τρία χαρακτηριστικά της που θα θέλατε να είχαν σχεδιαστεί διαφορετικά. Για ποιούς λόγους νομίζετε ότι έχουν σχεδιαστεί με τον τρόπο που υπάρχουν στη γλώσσα; Πώς θα τα τροποποιούσατε αν είχατε τη δυνατότητα να ξανασχεδιάσετε τη γλώσσα από την αρχή; Θα υπήρχαν κάποιες συνέπειες από αυτόν τον επανασχεδιασμό στη δυσκολία υλοποίησης της γλώσσας ή στην επίδοση των προγραμμάτων; Δικαιολογήστε συνοπτικά αλλά με επάρκεια όλες τις απαντήσεις σας. Δε χρειάζεται να γράψετε έκθεση ιδεών, αλλά η βαθμολογία σας θα εξαρτηθεί από την ορθότητα και τη σαφήνεια των απαντήσεών σας στις παραπάνω ερωτήσεις.

7. Προγραμματισμός σε Prolog (0.75 + 1 = 1.75 βαθμοί)

ΠΡΟΣΟΧΗ: Στην ερώτηση αυτή δε μπορείτε να θεωρήσετε ως δεδομένο κάποιο κατηγόρημα (π.χ. `member/2`, `append/3`, ...) της βιβλιοθήκης της Prolog. Αν το χρειάζεστε, θα πρέπει να το γράψετε.

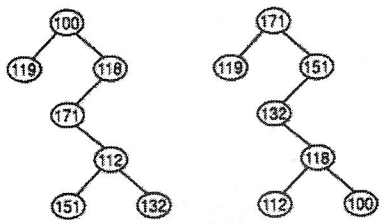
α) Το πρόβλημα των μαγικών τετραγώνων μάλλον είναι γνωστό στους περισσότερους από σας. Έστω το παρακάτω 3x3 τετράγωνο με τιμές V1 με V9 οι οποίες στην Prolog μπορούν εύκολα να αναπαρασταθούν με χρήση της λίστας [V1,V2,V3,V4,V5,V6,V7,V8,V9].

V1	V2	V3
V4	V5	V6
V7	V8	V9

Το τετράγωνο είναι "μαγικό" αν δώσουμε στα V1 με V9 τιμές από το 1 έως το 9 τέτοιες ώστε τα αθροίσματα των τριών γραμμών (V1+V2+V3, V4+V5+V6, και V7+V8+V9), τα αθροίσματα των τριών στηλών (V1+V4+V7, V2+V5+V8, και V3+V6+V9) και τα αθροίσματα των δύο διαγωνίων (V1+V5+V9 και V3+V5+V7) να είναι τα ίδια. Γράψτε σε Prolog τον ορισμό του κατηγορήματος `magic(L).` Όταν καλείται με το `L` μεταβλητή, το κατηγόρημά σας θα πρέπει να μπορεί να παράγει τη μία μετά την άλλη όλες τις λύσεις.

β) Ένα δυαδικό δένδρο είναι είτε κενό ή αποτελείται από κόμβους που περιέχουν ένα στοιχείο ως δεδομένο (π.χ. έναν ακέραιο) και δύο υποδένδρα τα οποία με τη σειρά τους είναι και αυτά δυαδικά δένδρα. Αφού πρώτα περιγράψετε σύντομα κάποιον κατάλληλο τρόπο για να αναπαραστήσετε τα δυαδικά δένδρα σε Prolog, να γράψετε το κατηγόρημα **`bfs_level(Tr,L)`** το οποίο είναι αληθές εάν η λίστα **`L`** αποτελείται από στοιχεία τα οποία βρίσκονται στο ίδιο βάθος (επίπεδο) στο δένδρο **`Tr`**. Για παράδειγμα, στα δύο παρακάτω δυαδικά δένδρα, έστω **`T1`** και **`T2`**, το κατηγόρημά σας θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω:

```
?- bfs_level(T1, L1).
L1 = [100] ;
L1 = [119,118] ;
L1 = [171] ;
L1 = [112] ;
L1 = [151,132] ;
false
```




```
?- bfs_level(T2, L2).
L2 = [171] ;
L2 = [119,151] ;
L2 = [132] ;
L2 = [118] ;
L2 = [112,100] ;
false
```

Σε ένα κενό δυαδικό δένδρο, το κατηγόρημά σας θα πρέπει (κανονικά) να αποτυγχάνει. Αλλά αν για κάποιο λόγο σας βολεύει να επιτυγχάνει με την κενή λίστα, δε θα το κάνουμε θέμα...

Καλή επιτυχία!