

Επώνυμο:
Όνομα:
Αριθμός Μητρώου:

1	0.8	
2	1.2	
3	1	
4	2	
5	1	
6	1	
7	1	
Σ	8	

## Τελικό Διαγώνισμα (Κανονική Εξέταση)

Το διαγώνισμα αυτό έχει ερωτήσεις 8 βαθμών συνολικά.

### 1. Γραμματικές (0.2 + 0.6 = 0.8 βαθμοί)

Έστω η παρακάτω γραμματική χωρίς συμφραζόμενα για μία γλώσσα ταυτόχρονου προγραμματισμού. Ο τελεστής `||` (παράλληλη εκτέλεση) έχει τη μικρότερη προτεραιότητα, έπεται ο τελεστής `;` (ακολουθιακή εκτέλεση) και στη συνέχεια ο τελεστής **unless** που έχει τη μεγαλύτερη προτεραιότητα. Όλοι οι τελεστές είναι αριστερά προσεταιριστικοί. Τα **a**, **b**, **c** παριστάνουν ατομικές ενέργειες. Τα άγκιστρα χρησιμεύουν για την ομαδοποίηση των εντολών.

```
<Stmt> ::= <Atom> | { <Stmt> } | <Stmt> <Op> <Stmt>
<Atom> ::= a | b | c
<Op>    ::= || | ; | unless
```

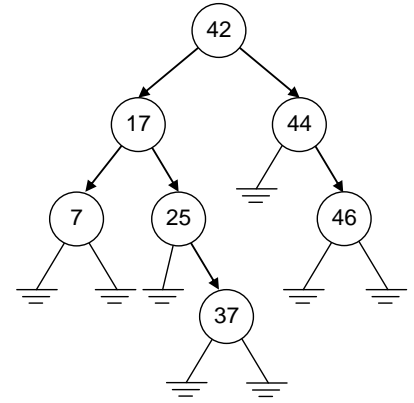
α) Δείξτε ότι η παραπάνω γραμματική είναι διφορούμενη.

β) Τροποποιήστε τη γραμματική ώστε να μην είναι διφορούμενη και οι τελεστές να έχουν τη σωστή προτεραιότητα και προσεταιριστικότητα.

## 2. Δένδρα σε ML και Prolog (0.6 + 0.6 = 1.2 βαθμοί)

Δίνεται ένα δένδρο δυναμικής αναζήτησης με πληροφορία ακέραιους αριθμούς. Δίνεται επίσης ένας ακέραιος **K**. Ζητείται η μέγιστη τιμή που περιέχεται στο δένδρο η οποία δεν υπερβαίνει το **K**. Προσέξτε ότι μπορεί να μην υπάρχει τέτοια τιμή. Για παράδειγμα, για το δένδρο του σχήματος και για **K=30**, η ζητούμενη τιμή είναι 25.

- α) Ορίστε σε ML έναν κατάλληλο τύπο δεδομένων `'a tree` για τα δυαδικά δένδρα. Στη συνέχεια, γράψτε μία κομψή και αποδοτική συνάρτηση `floor` που να κάνει τα παραπάνω. Ο τύπος της πρέπει να είναι: `int tree -> int -> int option`. (Θυμηθείτε: `datatype 'a option = NONE | SOME of 'a`)



- β) Περιγράψτε πώς θα μπορούσατε να αναπαραστήσετε δυαδικά δένδρα ως όρους της Prolog. Στη συνέχεια, γράψτε ένα κομψό και αποδοτικό κατηγορημα `floor(T, K, F)` το οποίο να επιτυγχάνει αν και μόνο αν η τιμή **F** είναι η μέγιστη που περιέχεται στο δένδρο **T** και δεν υπερβαίνει το **K**.

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

Συμπληρώστε τους τύπους των παρακάτω συναρτήσεων:

```
fun foo x y      = x * y x + 1
fun bar x y      = x y + y 42
fun doh (x, y)   = y (hd x) x
fun ugh x y z    = x y (z y)
```

	Τύπος
foo	
bar	
doh	
ugh	

### 4. Ερωτήσεις πολλαπλής επιλογής (8 \* 0.25 = 2 βαθμοί)

Κάθε λάθος απάντηση αφαιρεί 0.1 βαθμό από αυτό το θέμα.

Έστω το διπλανό πρόγραμμα σε Python.

α1) Τι θα εκτυπώσει η κλήση q1 ();

- A. 1 2 1                      B. 1 2 2  
Γ. 1 2 3                      Δ. άλλο

α2) Τι θα εκτυπώσει η κλήση q2 ();

- A. 1 1 1                      B. 1 2 2  
Γ. 3 2 2                      Δ. άλλο

<pre>a = b = 1 c = [b] def f(x):     print(x)     a = 3     x = 2     print(x) def q1():     f(a)     print(a)</pre>	<pre>def g(x):     global b     b = 3     print(x[0])     x[0] = 2     print(c[0]) def q2():     g(c)     print(c[0])</pre>
--	---

β) Έστω το διπλανό πρόγραμμα σε Prolog.  
Πόσα ζεύγη απαντήσεων (X, Y) θα δώσουν  
οι παρακάτω ερωτήσεις-στόχοι;

β1) qa(X, Y) .

- A. 3      B. 6      Γ. 9      Δ. άλλο

β2) qb(X, Y) .

- A. 1      B. 2      Γ. 3      Δ. άλλο

β3) qc(X, Y) .

- A. 1      B. 2      Γ. 3      Δ. άλλο

```
p(17) .
p(42) .
p(7) .

qa(X,Y) :- p(X) , p(Y) , X >= Y.
qb(X,Y) :- p(X) , ! , p(Y) , X >= Y.
pc(X)    :- p(X) , !.
qc(X,Y)  :- pc(X) , pc(Y) , X >= Y.
```

γ) Η υλοποίηση μίας γλώσσας προγραμματισμού αποθηκεύει στο σωρό κατά την εκτέλεση των προγραμμάτων πολλά μικρά αντικείμενα (2 λέξεων), τα οποία ζουν πολύ λίγο. Ποιο μηχανισμό συλλογής σκουπιδιών θα προτεινάτε, αν το κύριο μέλημα ήταν η ταχύτητα συλλογής;

- A. Μαρκάρισμα και σκούπισμα (mark and sweep)                      B. Αντιγραφής (copying)  
Γ. Μέτρηση αναφορών (reference counting)                      Δ. Οποιονδήποτε από τους A και B

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

δ1) Αν η γλώσσα υλοποιεί στατική αποστολή μεθόδων (static dispatch), το πρόγραμμα θα εκτυπώσει:

A. 1 2 1 2 3 4

B. 1 2 1 4 3 4

Γ. 1 2 3 2 3 4

Δ. 1 2 3 4 3 4

δ2) Αν η γλώσσα υλοποιεί δυναμική αποστολή μεθόδων (dynamic dispatch), το πρόγραμμα θα εκτυπώσει:

A. 1 2 1 2 3 4

B. 1 2 1 4 3 4

Γ. 1 2 3 2 3 4

Δ. 1 2 3 4 3 4

```
class A {  
    foo() { print(1); bar(); }  
    bar() { print(2); }  
}  
  
class B extends A {  
    foo() { print(3); bar(); }  
    bar() { print(4); }  
}  
  
main() {  
    A a = new A; a.foo();  
    a = new B; a.foo();  
    B b = new B; b.foo();  
}
```

## 5. Πέρασμα παραμέτρων (0.5 + 0.5 = 1 βαθμός)

Έστω μία γλώσσα συναρτησιακού προγραμματισμού, για την οποία δεν είμαστε σίγουροι με ποιο τρόπο περνούν οι παράμετροι των συναρτήσεων. Μπορεί αυτό να γίνεται είτε **κατ' όνομα** (call by name), είτε **κατ' ανάγκη** (call by need), αλλά δε γνωρίζουμε ποιο από τα δύο.

α) Γράψτε ένα πρόγραμμα που θα σας επιτρέψει να διαπιστώσετε με ποιο τρόπο περνούν οι παράμετροι. Εξηγήστε σύντομα τι περιμένετε σε κάθε περίπτωση. Μπορείτε να υποθέσετε ότι η γλώσσα έχει παρενέργειες (π.χ. ανάθεση σε μεταβλητές, εκτύπωση, κ.λπ.)

β) Όπως το (α) αλλά η γλώσσα είναι αγνή συναρτησιακή, επομένως δεν υπάρχουν παρενέργειες.

## 6. Προγραμματισμός σε ML (1 βαθμός)

Να γραφεί σε ML μία κομψή και αποδοτική συνάρτηση `allsubseq` η οποία να δέχεται ως παράμετρο μία λίστα `L` και να επιστρέφει μία λίστα αποτελούμενη από όλες τις υποακολουθίες της `L`. (Η λίστα `S` είναι υποακολουθία της λίστας `L` αν η `S` προκύπτει από την `L` με την αφαίρεση μηδενός ή περισσότερων στοιχείων.) Η σειρά με την οποία θα εμφανίζονται οι υποακολουθίες στο αποτέλεσμα δεν είναι σημαντική. Παραδείγματα δίνονται παρακάτω:

```
- allsubseq [1,2,3];  
val it = [[], [1], [2], [1,2], [3], [1,3], [2,3], [1,2,3]] : int list list  
  
- allsubseq ["one", "two"];  
val it = [[], ["one"], ["two"], ["one", "two"]] : string list list
```

Φροντίστε ο κώδικάς σας να είναι κατανοητός και ευανάγνωστος! Προσθέστε σχόλια αν χρειάζονται!

## 7. Προγραμματισμός σε Prolog (1 βαθμός)

Να γραφεί σε Prolog ένα κομψό και αποδοτικό κατηγορήμα `incsubseq(L,K,S)`, το οποίο να αληθεύει αν και μόνο αν η λίστα **S** είναι μία υποακολουθία της λίστας **L** μήκους **K** και τα στοιχεία της είναι σε αύξουσα σειρά. (Η λίστα **S** είναι υπακολουθία της λίστας **L** αν η **S** προκύπτει από την **L** με την αφαίρεση μηδενός ή περισσότερων στοιχείων.) Η λύση σας πρέπει να συμπεριφέρεται ως εξής:

```
?- incsubseq([2,1,4,5,4], 3, S).      ?- incsubseq([3,2,1], 2, S).
S = [2,4,5] ;                          false.
S = [2,4,4] ;
S = [1,4,5] ;
S = [1,4,4] ;
false.                                ?- incsubseq([3,4,1,2], 3, S).
                                         false.
```

Φροντίστε ο κώδικάς σας να είναι κατανοητός και ευανάγνωστος! Προσθέστε σχόλια αν χρειάζονται!

**Καλή επιτυχία!**