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

Εαρινό 2007

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Τελικό Διαγώνισμα (Κανονική Εξέταση)

Το διαγώνισμα αυτό έχει ερωτήσεις 8 βαθμών συνολικά και μια προαιρετική ερώτηση 0.5 βαθμών.

1. Προγραμματισμός σε ML (0.8 + 0.2 βαθμοί)

Έστω ότι **f** και **g** είναι συναρτήσεις και **lst** είναι μια λίστα από n ≥ 0 τιμές: **[v_1,v_2,..., v_n].** Γράψτε μια συνάρτηση **munge(f,g,lst)** σε ML που να υπολογίζει την παρακάτω λίστα με 2*n τιμές: **[f(v_1),g(v_n),f(v_2),g(v_{n-1}),...,f(v_n),g(v_1)].** Ποιος είναι ο τύπος της **munge**;

2. Συμπερασμός τύπων στην ML (0.5+0.25+0.25 βαθμοί)

α) Ποιος είναι ο τύπος της παρακάτω συνάρτησης στην ML;

```
fun r(f,[a,b]) = f(a,b)
  | r(f,u::x::y::z) = f(u,r(f,x::y::z));
```

β) Ο διερμηνέας της ML θα βγάλει κάποιο προειδοποιητικό μήνυμα (warning) για την παραπάνω συνάρτηση. Τι μήνυμα είναι αυτό και τι σημαίνει;
γ) Τι ακριβώς κάνει η συνάρτηση αυτή; Αν δε μπορείτε να το περιγράψετε με λόγια, δώστε κάποια παραδείγματα κλήσης της συνάρτησης και τα αποτελέσματα των κλήσεων.

3. Εμβέλεια ονομάτων (1+1 βαθμοί)

α) Οι περισσότερες γλώσσες προγραμματισμού (π.χ. η C, η C++ και η Java) επιτρέπουν τη φωλιασμένη ορατότητα (nested visibility) ονομάτων. Με άλλα λόγια, σε μια σειρά από φωλιασμένες εμβέλειες ονομάτων, ο πιο κοντινός (πιο εσωτερικός) ορισμός ενός ονόματος είναι αυτός που πάντα χρησιμοποιείται, ακόμα και εάν το όνομα αυτό έχει και άλλους ορισμούς με περικλείουσες εμβέλειες.
Τι είδους πλεονεκτήματα έχει η φωλιασμένη ορατότητα ονομάτων για μια γλώσσα; Υπάρχουν κάποια μειονεκτήματα; Εάν ναι, πως μπορούμε να τα ξεπεράσουμε;
β) Στις υλοποιήσεις της φωλιασμένης ορατότητας, χρησιμοποιείται πάντα ο πιο κοντινός ορισμός κάποιου ονόματος ακόμα και εάν ο ορισμός αυτός οδηγεί σε «προφανή» σφάλματα τύπου. Για παράδειγμα, εάν δούμε μια αναφορά **a[1]** σ' έναν πίνακα, θα χρησιμοποιήσουμε τον κοντινότερο ορισμό του **a** ακόμα και εάν ο ορισμός αυτός δηλώνει ότι το **a** είναι κάτι διαφορετικό από πίνακα. Φανταστείτε ότι τροποποιούμε τον κανόνα φωλιασμένης εμβέλειας έτσι ώστε να χρησιμοποιούμε πάντα τον κοντινότερο ορισμό *που όμως είναι έγκυρος*. Με άλλα λόγια τροποποιούμε τον κανόνα ώστε να επιτρέπεται να αγνοήσουμε δηλώσεις που είναι προφανώς λάθος (όπου λάθος σημαίνει διαφορετική από αυτή που ψάχνουμε στο συγκεκριμένο περιβάλλον χρήσης του ονόματος).
Η συγκεκριμένη αλλαγή στον ορισμό της φωλιασμένης εμβέλειας έχει κάποια μειονεκτήματα; Υπάρχουν περιπτώσεις που μπορεί να οδηγήσει σε ασαφές ή σε διφορούμενο αποτέλεσμα;

4. Αντικειμενοστρεφής προγραμματισμός (0.5+0.5 βαθμοί)

α) Ποια είναι η διαφορά μεταξύ μιας αφηρημένης κλάσης (abstract class) στη Java και μιας διαπροσωπείας (interface);
β) Τι σημαίνει για μια μέθοδο της Java να έχει δηλωθεί ως final;

5. Πέρασμα παραμέτρων (2 βαθμοί)

Έστω το παρακάτω πρόγραμμα γραμμένο στη σύνταξη της γλώσσας C:

```
void swap(int a, int b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
}

void main() {
  int value = 2, list[5] = {1,3,5,7,9};
  swap(value, list[0]);
  swap(list[0], list[1]);
  swap(value, list[value]);
}
```

Στους τέσσερις πίνακες που υπάρχουν στο «φύλλο απαντήσεων» αυτού του διαγωνίσματος βάλτε τις τιμές που θα έχουν οι μεταβλητές **value** και **list** *έπειτα* από την κάθε μία από τις τρεις κλήσεις της **swap** αν το πέρασμα παραμέτρων γίνεται με:

α) κλήση κατά τιμή
β) κλήση κατ' αναφορά
γ) κλήση κατ' όνομα
δ) κλήση κατά τιμή-αποτέλεσμα

6. Λογικός προγραμματισμός (1 βαθμός)

Η τρίλιζα είναι ένα παιχνίδι που παίζεται σ' ένα τετράγωνο 3x3 στο οποίο οι δύο παίκτες σημειώνουν διαδοχικά τους χαρακτήρες **x** και **o**. Όταν τρία **x** βρεθούν στην ίδια γραμμή, στήλη ή διαγώνιο, ο **x** παίκτης κερδίζει. Έστω ότι αναπαριστούμε στην Prolog την τρίλιζα με μια λίστα από τρείς υπολίστες. Κάθε μία από τις υπολίστες έχει τρία στοιχεία και τα στοιχεία αυτά μπορεί να είναι είτε κάποιο **x**, **o**, ή κάποιο **b** που υποδηλώνει ότι το τετράγωνο είναι κενό. Για παράδειγμα, η τρίλιζα:

x		o
o	x	x
o		x

αναπαρίσταται ως: **[[x,b,o],[o,x,x],[o,b,x]].**
Να γραφεί ένα πρόγραμμα σε Prolog που να ορίζει τη σχέση **winner(L)** όπου **L** είναι η μια λίστα από λίστες με την παραπάνω αναπαράσταση της τρίλιζας η οποία ήδη έχει κάποιες τιμές. Το κατηγόρημα **winner/1** θα πρέπει να επιτυγχάνει εάν η τρίλιζα **L** αναπαριστά μια θέση στην οποία ο παίκτης **x** έχει ήδη κερδίσει ή κερδίζει με το μετασχηματισμό ενός μόνο κενού **(b)** σε **x**. Με άλλα λόγια το κατηγόρημα **winner/1** πρέπει να αναγνωρίζει τρίλιζες στις οποίες ο παίκτης **x** είτε έχει ήδη κερδίσει ή κερδίζει στην αμέσως επόμενη κίνηση. Τα κατηγορήματά σας πρέπει να έχουν σχόλια που εξηγούν τι κάνουν.

Φύλλο Απαντήσεων

Ονοματεπώνυμο: _____ ***Α.Μ.*** _____

5. Πέρασμα παραμέτρων (0.5 βαθμοί το κάθε υπο-ερώτημα)

α) Κλήση κατά τιμή	**value**	**list**
`swap(value, list[0])`		
`swap(list[0], list[1])`		
`swap(value, list[value])`		

β) Κλήση κατ' αναφορά	**value**	**list**
`swap(value, list[0])`		
`swap(list[0], list[1])`		
`swap(value, list[value])`		

γ) Κλήση κατ' όνομα	**value**	**list**
`swap(value, list[0])`		
`swap(list[0], list[1])`		
`swap(value, list[value])`		

δ) Κλήση κατά τιμή-αποτέλεσμα	**value**	**list**
`swap(value, list[0])`		
`swap(list[0], list[1])`		
`swap(value, list[value])`		

7. Επιπλέον ερώτηση bonus (0.5 βαθμοί)

Σχεδιάστε τη μέγιστη διαδρομή που μπορεί να κάνει το φιδάκι ο Διαμαντής (Άσκηση 2) στο παρακάτω πάτωμα:

20	2	3	45	3	4	93	5
2	31	62	3	8	5	7	9
1	4	83	21	28	7	5	29
5	2	9	4	20	1	73	7
7	8	7	5	9	8	1	2
21	93	82	72	3	2	1	8
22	9	6	8	4	93	8	1
94	1	24	8	21	2	9	21

Καλή επιτυχία!