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

Εαρινό 2016

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: ??/7/2016, 23:59:59

Δίκαιες μοιρασιές (0.25 βαθμοί)

Το πρόβλημα με τις δίκαιες μοιρασιές είναι γνωστό από την πρώτη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του προβλήματος σε Prolog. Επειδή τα συστήματα Prolog δεν τρέχουν native code, ο χρονικός περιορισμός για την άσκηση αυξάνεται σε ένα λεπτό αντί για 10 secs. Το πρόγραμμά σας θα πρέπει να περιέχει ένα κατηγόρημα **`fair_parts/2`** το οποίο θα πρέπει να επιστρέφει στο δεύτερό του όρισμα μια λίστα από λίστες από αριθμούς που δείχνει πως θα γίνει η μοιρασιά με τον τρόπο που ορίζει ως «δίκαιη» η συγκεκριμένη άσκηση. Για το δεύτερο και το τρίτο παράδειγμα της εκφώνησης της άσκησης, το κατηγόρημά σας θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω[1].

```
?- fair_parts('objects2.txt', Parts).
Parts = [[42],[42,42],[42,42]] ;
false.
?- fair parts('objects3.txt', Parts).
Parts = [[10,20,30,40,50],[60,70],[80,90]] ;
false.
```

Στην ιστοσελίδα του μαθήματος μπορείτε να βρείτε ένα πρόγραμμα Prolog που διαβάζει αρχεία με δεδομένα εισόδου αυτής της άσκησης και σας τα επιστρέφει σε μια δομή δεδομένων την οποία μπορείτε είτε να τη χρησιμοποιήσετε ως έχει ή να την τροποποιήσετε κατάλληλα για τις ανάγκες της λύσης σας.

Ισοζυγισμένα βάρη (0.25 βαθμοί)

Το πρόβλημα με τα ισοζυγισμένα βάρη σας είναι γνωστό από τη δεύτερη σειρά ασκήσεων. Το ζητούμενο είναι να γραφεί η λύση του προβλήματος σε Prolog. Το κατηγόρημα **`balance/4`**, που το πρόγραμμά σας θα πρέπει να περιέχει, δέχεται στα δύο πρώτα του ορίσματα τους αριθμούς *N* και *W* και ενοποιεί τα δύο τελευταία του ορίσματα με δύο ταξινομημένες λίστες που δείχνουν το πως πρέπει να τοποθετηθούν τα αντικείμενα στις δύο μεριές της ζυγαριάς έτσι ώστε να είναι ισοβαρείς, ή αποτυγχάνει αν δεν υπάρχει κάποιος τέτοιος τρόπος. Παραδείγματα αντίστοιχα της εκφώνησης της δεύτερης σειράς φαίνονται παρακάτω.

```
?- balance(10, 2, L, R).
L = [1]
R = [2].
?- balance(10, 5, L, R).
L = [1,2]
R = [3].
```

```
?- balance(13, 4, L, R).
L = []
R = [1,2].
?- balance(10, 5, L, R).
false.
```

Όπως και στην προηγούμενη άσκηση, το όριο του χρόνου εκτέλεσης αυξάνεται σε ένα λεπτό.

[1] Σε όλα τα παραδείγματα αυτής της σειράς ασκήσεων, ανάλογα με το σύστημα Prolog που θα χρησιμοποιήσετε, στις περιπτώσεις που υπάρχει κάποια λύση, η γραμμή με το **`false.`** μπορεί να λέει **`fail.`** ή **`no`** ή μπορεί να μην τυπώνεται (που μάλλον είναι το καλύτερο διότι δείχνει ότι η εκτέλεση του κατηγορήματός σας είναι ντετερμινιστική).

Ντόμινο στο τετράγωνο (0.25+0.25 = 0.5 βαθμοί)

Τα πλακίδια του ντόμινο είναι ορθογώνια χωρισμένα στη μέση σε δύο τετράγωνα και έχουν από μηδέν έως έξι τελείες στις δύο μεριές τους. Είναι συνολικά 28 και, για να μη ζωγραφίζουμε τελείες, θα τα αναπαραστήσουμε με δυάδες (0,0), (0,1), (0,2), ..., (0,6), (1,1), (1,2), ..., (5,5), (5,6), (6,6) από ακεραίους από το 0 έως το 6 σε αυτήν την άσκηση.

Σας δίνεται ως είσοδος ένα αρχείο το οποίο μέσα του περιέχει ένα ορθογώνιο με 8 στήλες και 7 γραμμές το οποίο έχει 56 ακεραίους, επίσης από το 0 έως το 6. Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε Prolog και ένα άλλο σε όποια γλώσσα επιθυμείτε από τις C/C++, ML ή Java) τα οποία να επιστρέφουν (στην Prolog και στην ML) ή να τυπώνουν (στις C/C++ και Java) το πλήθος των διαφορετικών τρόπων με τους οποίους μπορούμε να τοποθετήσουμε τα 28 πλακίδια του ντόμινο στο συγκεκριμένο ορθογώνιο ώστε να καλύπτονται όλοι οι 56 αριθμοί του. Δύο παραδείγματα εισόδου φαίνονται παρακάτω. Φυσικά, το πρόγραμμά σας θα πρέπει να επιστρέφει/τυπώνει 0 αν δεν υπάρχει κάποιος τέτοιος τρόπος.

Περιορισμοί: Όρια χρόνου: 1 λεπτό (για Prolog) και 10 δευτερόλεπτα (για τις άλλες γλώσσες). Όριο μνήμης: 1.5GB.

Το πρόγραμμά σας σε Prolog πρέπει να ορίζει ένα κατηγόρημα **dominos/2** που να συμπεριφέρεται ως εξής:

```
?- dominos('square1.txt', N).
N = 1.
?- dominos('square2.txt', N).
N = 18.
```

Το πρόγραμμά σας σε SML/NJ πρέπει να έχει τη συμπεριφορά που φαίνεται παρακάτω:

```
- dominos "square2.txt";
val it = 18 : int
```

Το πρόγραμμά σας σε Java, C, C++, MLton ή OCaml πρέπει να δουλεύει όπως φαίνεται παρακάτω:

```
$ java Dominos square2.txt
18
$ ./a.out square2.txt
18
```

Τα δύο αρχεία εισόδου είναι:

```
> cat square1.txt
5 3 1 0 0 1 6 3
0 2 0 4 1 2 5 2
1 5 3 5 6 4 6 4
0 5 0 2 0 4 6 2
4 5 3 6 0 6 1 1
2 3 5 3 4 4 5 3
2 1 1 6 6 2 4 3
```

```
> cat square2.txt
0 3 0 2 2 0 2 3
1 5 6 5 5 1 2 2
3 4 1 4 5 4 4 4
6 6 1 0 5 2 3 0
4 0 3 2 4 1 6 0
1 4 1 5 6 6 3 0
1 2 6 5 5 6 3 3
```

Περαιτέρω οδηγίες για την άσκηση

- Μπορείτε να δουλέψετε σε ομάδες το πολύ δύο ατόμων. Μπορείτε αν θέλετε να σχηματίσετε διαφορετική ομάδα σε σχέση με τις προηγούμενες σειρές ασκήσεων – οι ομάδες στο σύστημα υποβολής είναι έτσι και αλλιώς καινούργιες για κάθε σειρά.

- Δεν επιτρέπεται να μοιράζεστε τα προγράμματά σας με συμφοιτητές εκτός της ομάδας σας ή να τα βάλετε σε μέρος που άλλοι μπορούν να τα βρουν (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοσελίδες συζητήσεων, …). Σε περίπτωση που παρατηρηθούν «περίεργες» ομοιότητες σε προγράμματα, ο βαθμός των εμπλεκόμενων φοιτητών στις ασκήσεις γίνεται αυτόματα μηδέν ανεξάρτητα από το ποια ομάδα... «εμπνεύστηκε» από την άλλη.

- Μπορείτε να χρησιμοποιήσετε «βοηθητικό» κώδικα (π.χ. κάποιο κώδικα που διαχειρίζεται κάποια δομή δεδομένων) που βρήκατε στο διαδίκτυο στα προγράμματά σας, με την προϋπόθεση ότι το πρόγραμμά σας περιέχει σε σχόλια την παραδοχή για την προέλευση αυτού του κώδικα και ένα σύνδεσμο σε αυτόν.

- Τα προγράμματα σε Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog (6.6.6), GNU Prolog (1.3.0) ή YAP (6.2.2).

- Τα προγράμματα στην άλλη γλώσσα μπορεί να είναι σε C/C++, ML (SML/NJ v110.76 ή MLton 20100608 ή Objective Caml version 4.01.0) ή σε Java. Το σύστημα ηλεκτρονικής υποβολής επιτρέπει να επιλέξετε μεταξύ αυτών των γλωσσών και των υλοποιήσεων της ML.

- Ο κώδικας των προγραμμάτων σε C/C++ και ML πρέπει να είναι σε ένα αρχείο. Ο κώδικας των προγραμμάτων σε Java μπορεί να βρίσκεται σε περισσότερα του ενός αρχείου αλλά θα πρέπει να μπορεί να μεταγλωττιστεί χωρίς προβλήματα με τον Java compiler με την εντολή **`javac Dominos.java`**

- Η υποβολή των προγραμμάτων θα γίνει ηλεκτρονικά όπως και στην προηγούμενη άσκηση. Θα υπάρξει σχετική ανακοίνωση μόλις το σύστημα υποβολής καταστεί ενεργό. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο εκτός από τη ζητούμενη διότι δε θα γίνουν δεκτά από το σύστημα υποβολής.