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

Εαρινό 2010

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 21/7/2010, 14:30

Λείοι αριθμοί (0.25 βαθμοί)

Το πρόβλημα με τους λείους αριθμούς είναι γνωστό από την πρώτη σειρά ασκήσεων. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του προβλήματος σε Prolog. Κάποια παραδείγματα, αντίστοιχα αυτών της πρώτης άσκησης, από τη χρήση του κυρίου κατηγορήματος **smooth/4** που το πρόγραμμά σας πρέπει να περιέχει φαίνονται παρακάτω.

```
?- smooth(2, 1, 42, Smooths).
Smooths = 6 ;
fail.
?- smooth(3, 42, 420, Smooths).
Smooths = 17 ;
fail.
?- smooth(7, 33, 192, Smooths).
Smooths = 42 ;
fail.
```

Master Mind (0.25 βαθμοί)

Το πρόβλημα του Master Mind είναι γνωστό από τη δεύτερη σειρά ασκήσεων. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του προβλήματος σε Prolog. Κάποια παραδείγματα, αντίστοιχα αυτών της δεύτερης άσκησης, από τη χρήση του κυρίου κατηγορήματος **mastermind/3** που το πρόγραμμά σας πρέπει να περιέχει φαίνονται παρακάτω.

```
?- mastermind(4, [move('3157',1,2),move('1350',2,1),
                  move('6120',0,2),move('2381',3,0)], MinCode).
MinCode = '2351' ;
fail.
?- mastermind(7, [], MinCode).
MinCode = '0000000' ;
fail.
```

Όταν δεν υπάρχει κάποιος αριθμός που να αντιστοιχεί σε κάποιον ελάχιστο κωδικό, το κατηγόρημα **mastermind/3** πρέπει να αποτυγχάνει (δηλαδή να μην επιστρέφει κάποια απάντηση).

Όπως φαίνεται και στα παραπάνω παραδείγματα, οι κώδικες αναπαριστάνονται με Prolog άτομα. Στο πρόγραμμά σας πιθανώς να σας βοηθήσει το ενσωματωμένο κατηγόρημα **atom_chars/2** το οποίο μετατρέπει ένα άτομο στη λίστα των χαρακτήρων από τους οποίους αποτελείται και αντίστροφα όπως φαίνεται και στις κλήσεις

```
?- atom_chars('42', L).
L = ['4', '2'].
?- atom_chars(A, ['4', '2']).
A = '42'.
```

Ανάλογο του **atom_chars/2** είναι και το ενσωματωμένο κατηγόρημα **atom_codes/2**.

Επίσης, ανάλογα με τον αλγόριθμο που θα χρησιμοποιήσετε, μπορεί να σας φανούν χρήσιμοι οι τελεστές σύγκρισης όρων της Prolog οι οποίοι δουλεύουν για κάθε "τύπο" όρων, όπως φαίνεται παρακάτω.

```
?- '4711' @< '1234'.
fail.
?- '3157' @< '4242'.
true.
?- ['2','1'] @=< ['4','2'].
true.
```

Φυσικά όμως, δεν υπάρχει καμία απολύτως υποχρέωση το πρόγραμμά σας να χρησιμοποιεί τους παραπάνω τελεστές και κατηγορήματα.

S&M (0.25+0.25 βαθμοί)

Σας δίνεται ένα σύνολο από δεκαδικούς αριθμούς με το ίδιο μήκος οι οποίοι είναι γραμμένοι ο ένας κάτω από τον άλλον όπως στο αριστερό μέρος της παρακάτω εικόνας. Παρατηρήστε ότι κάποιοι αριθμοί μπορεί να ξεκινούν με μηδενικά.

12345	17654
23456	73456
00934	90934
76543	26543
30303	60303

Είναι εύκολο να υπολογίσουμε το άθροισμά τους: **143581**. Όμως επιτρέπεται να κάνουμε και δύο "διαστροφές" σε αυτούς τους αριθμούς: μπορούμε να διαστρέψουμε όλα τα ψηφία μιας γραμμής ή μπορούμε να διαστρέψουμε όλα τα ψηφία μιας στήλης. Διαστροφή ενός ψηφίου **d** σημαίνει να το αντικαταστήσουμε από το ψηφίο **9-d**. Οπότε είναι εύκολο να δείτε ότι οι αριθμοί στο δεξί μέρος της εικόνας προκύπτουν από το αριστερό με διαστροφή της πρώτης γραμμής και της πρώτης στήλης. Τώρα το άθροισμά τους έχει αλλάξει και είναι **268890**.

Προφανώς, ξεκινώντας από ένα σύνολο αριθμών και κάνοντας όσες διαστροφές γραμμών και στηλών θέλουμε, υπάρχει κάποιο μέγιστο άθροισμα που μπορούμε να πετύχουμε. Φυσικά δεν έχει νόημα να εφαρμόσουμε κάποια διαστροφή στην ίδια γραμμή ή στήλη δύο συνεχόμενες φορές διότι έτσι καταλήγουμε στην αρχική γραμμή ή στήλη. Για κάθε σύνολο αριθμών, για να πετύχουμε αυτό το μέγιστο άθροισμα υπάρχει ένας ελάχιστος και ένας μέγιστος αριθμός διαστροφών που πρέπει να κάνουμε.

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε Prolog και ένα στη γλώσσα της αρεσκείας σας μεταξύ των C, ML ή Java) που να υπολογίζουν το μέγιστο αυτό άθροισμα και τον ελάχιστο και μέγιστο αριθμό διαστροφών που χρειάζονται για να πετύχουμε το άθροισμα αυτό.

Το πρόγραμμά σας σε Prolog πρέπει να ορίζει ένα κατηγόρημα που να δουλεύει ως εξής:

```
?- snm(['12345','23456','00934','76543','30303'], Sum, Min, Max).
Sum = 409604
Min = 4
Max = 6 ;
fail.
```

Επειδή οι αριθμοί στη λίστα εισόδου αναπαριστώνται ως άτομα, τα κατηγορήματα που αναφέρονται στην εκφώνηση του master mind μπορεί να σας φανούν χρήσιμα.

Το πρόγραμμά σας σε ML δέχεται τους αριθμούς εισόδου ως strings και πρέπει να έχει τη συμπεριφορά εισόδου/εξόδου που φαίνεται παρακάτω:

```
- snm ["12345","23456","00934","76543","30303"];
val it = (409604,4,6) : IntInf.int * int * int
```

Το πρόγραμμά σας σε Java (ή σε C) πρέπει να έχει τη συμπεριφορά εισόδου/εξόδου που φαίνεται παρακάτω (η πρώτη γραμμή εισόδου δείχνει το πλήθος των αριθμών στις επόμενες γραμμές):

```
    > java SnM.class
είσοδος
    5
    12345
    23456
    00934
    76543
    30303
έξοδος
    409604
    4
    6
```

Παρά την όλη διαστροφή της άσκησης :-), μπορείτε να υποθέσετε ότι η είσοδος του προγράμματος θα αποτελείται από αριθμούς ίδιου μήκους, οπότε το πρόγραμμά σας δε χρειάζεται να ελέγχει την ορθότητα της εισόδου.

Περαιτέρω οδηγίες για την άσκηση

- Μπορείτε να δουλέψετε σε ομάδες το πολύ 2 ατόμων (αλλά μπορείτε αν θέλετε να σχηματίσετε διαφορετική ομάδα σε σχέση με τις προηγούμενες ασκήσεις).
- Δεν επιτρέπεται να μοιράζεστε ασκήσεις με άλλους συμφοιτητές σας ή να βάλετε τις ασκήσεις σας σε μέρος που άλλοι μπορούν να τις βρουν εύκολα (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοτόπους συζητήσεων, …).
- Τα προγράμματα σε Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog, GNU Prolog ή YAP.
- Το πρόγραμμα στην άλλη γλώσσα μπορεί να είναι σε C, ML (SML/NJ ή MLton) ή Java.
  Η αποστολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του moodle, όπως και στις προηγούμενες ασκήσεις. Θα υπάρξει σχετική ανακοίνωση μόλις το submission site καταστεί ενεργό. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο διότι δε θα γίνουν δεκτά από το σύστημα στο οποίο θα υποβληθούν.