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

Εαρινό 2014

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 25/9/2014, 23:59:59

Εμπρός για νέους δρόμους ανάπτυξης! (0.25 βαθμοί)

Το πρόβλημα με τους δρόμους ανάπτυξης 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 **`dromoi/2`** το οποίο, αν υπάρχει λύση θα πρέπει να επιστρέφει στο δεύτερό του όρισμα τον ελάχιστο αριθμό ημερών που πρέπει να εργαστεί το συνεργείο, και να αποτυγχάνει αν δεν υπάρχει λύση. Με άλλα λόγια, για τα παραδείγματα της εκφώνησης της πρώτης άσκησης θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω[1].

```
?- dromoi('anaptyksi1.txt', Days).
Days = 2 ;
false.
?- dromoi('anaptyksi2.txt', Days).
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.

Κουβαδάκια στην παραλία... (0.25 βαθμοί)

Το πρόβλημα με τα κουβαδάκια στην παραλία είναι γνωστό από τη δεύτερη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο είναι να γραφεί η λύση του προβλήματος σε Prolog. Το κατηγόρημα **`kouvadakia/4`** που το πρόγραμμά σας πρέπει να περιέχει δέχεται τρεις ακεραίους στα πρώτα του ορίσματα και, αν υπάρχει λύση επιστρέφει μια λίστα από τα άτομα **`'01'`**, **`'02'`**, **`'12'`**, **`'21'`**, **`'10'`**, και **`'20'`**, αλλιώς το κατηγόρημά σας πρέπει να αποτυγχάνει. Παραδείγματα αντίστοιχα αυτών της δεύτερης σειράς φαίνονται παρακάτω.

```
?- kouvadakia(3, 4, 1, Plan).
Plan = ['02','21'].
?- kouvadakia(5, 7, 6, Plan).
Plan = ['02','21','10','21','02','21','10','21','02','21'].
?- kouvadakia(4, 2, 1, Plan).
false.
```

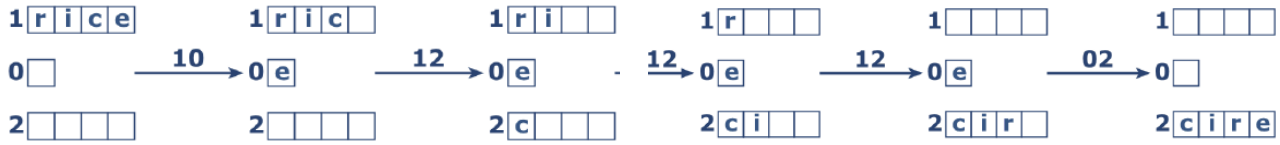
Όπως και στην προηγούμενη άσκηση, το όριο του χρόνου εκτέλεσης αυξάνεται σε ένα λεπτό.

[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).

Αναγραμματισμοί με δύο στοίβες... (0.25+0.25 = 0.5 βαθμοί)

Μας δίδονται δύο συμβολοσειρές που η μία είναι ένας αναγραμματισμός της άλλης. Ξεκινώντας από τη στοίβα 1 η οποία περιλαμβάνει την πρώτη λέξη έτσι ώστε το τελευταίο γράμμα της να είναι πάνω-πάνω στη στοίβα, μια δεύτερη στοίβα η οποία συμβολίζεται με 2 και αρχικά είναι κενή και έναν καταχωρητή που συμβολίζεται με 0 και χωράει ένα μόνο χαρακτήρα, αυτό που θέλουμε να κάνουμε είναι να μετακινήσουμε τα γράμματα της πρώτης λέξης στη δεύτερη στοίβα έτσι ώστε να σχηματιστεί εκεί η δεύτερη λέξη. Φυσικά θέλουμε να πραγματοποιήσουμε αυτή τη μετακίνηση με τις ελάχιστες δυνατές κινήσεις.

Η παρακάτω εικόνα δείχνει το τι συμβαίνει σε ένα παράδειγμα. Οι συμβολοσειρές **"rice"** και **"cire"** είναι αναγραμματισμοί η μία της άλλης.



Ξεκινάμε από τη συμβολοσειρά **"rice"** στην πρώτη στοίβα και θέλουμε να καταλήξουμε με τη συμβολοσειρά **"cire"** στη δεύτερη στοίβα. Μετακινούμε πρώτα το **e** στον καταχωρητή. Η κίνηση αυτή συμβολίζεται με 10 (από τη στοίβα 1 στον καταχωρητή 0). Μετά μετακινούμε το **c** από την πρώτη στη δεύτερη στοίβα και η μετακίνηση αυτή συμβολίζεται με 12. Νομίζω ότι πλέον η παραπάνω εικόνα είναι προφανής.

Αυτό που ζητάει η άσκηση είναι να γραφούν δύο προγράμματα (ένα σε Prolog και το άλλο σε όποια γλώσσα επιθυμείτε από τις C/C++, ML ή Java) τα οποία να παίρνουν ως είσοδο τις δύο συμβολοσειρές και να επιστρέφουν ως έξοδο την ακολουθία των ελάχιστων κινήσεων με τις οποίες μπορεί να γίνει η μετακίνηση είτε ως λίστα από άτομα (στην Prolog) ή ως συμβολοσειρές (στις άλλες γλώσσες) όπως φαίνεται παρακάτω.

Περιορισμοί: 1 ≤ **|συμβολοσειρά εισόδου|** ≤ 42. Όριο μνήμης: 4GB, χρόνου εκτέλεσης: 10 λεπτά (για Prolog), 1 λεπτό (για τις άλλες γλώσσες).

Το πρόγραμμά σας σε Prolog πρέπει να περιέχει ένα κατηγόρημα **anagrams/3** που να συμπεριφέρεται ως εξής:

```
?- anagrams("rice", "cire", Moves).
Moves = ['10','12','12','12','02'].
?- anagrams("anagram", "mragana", Moves).
Moves = ['12','10','12','02','12','12','12','12'].
?- anagrams("mirror", "mirorr", Moves).
Moves = ['12','12','10','12','12','02','10','21','21','21','21','21','02','12','12',
         '12','10','21','21','21','02','12','12','12','12'].
```

Το τελευταίο παράδειγμα μπορεί να λυθεί και με άλλους τρόπους. Παρατηρήστε όμως ότι το πρόγραμμα δίνει πάντα μια λύση και μετά τερματίζει την οπισθοδρόμηση.

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

```
- anagrams ("rice", "cire");
val it = "10-12-12-12-02" : string
```

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

```
$ java Anagrams rice cire
10-12-12-12-02
$ ./a.out rice cire
10-12-12-12-02
```

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.
- Τα προγράμματα σε Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog (6.4.1), GNU Prolog ή YAP.
- Τα προγράμματα στην άλλη γλώσσα μπορεί να είναι σε C/C++, ML (SML/NJ v110.74 ή MLton 20100608 ή Objective Caml version 3.11.2) 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 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 Anagrams.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.