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

Εαρινό 2009

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

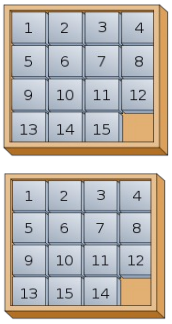
Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 16/7/2009, 23:59

Ρίξε μια ζαριά καλή... (0.25 βαθμοί)

Το πρόβλημα με τη ζαριά είναι γνωστό από τη δεύτερη σειρά ασκήσεων. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του προβλήματος σε Prolog. Κάποια παραδείγματα από τη χρήση του κυρίου κατηγορήματος **`zaria/4`** που το πρόγραμμά σας πρέπει να περιέχει φαίνονται παρακάτω. Παρατηρείστε ότι το κατηγόρημα πρέπει να αποτυγχάνει αν δεν υπάρχει λύση.

```
?- zaria(3, [(1,2),(2,1),(2,3),(3,2)], [3,5], Moves).
Moves = 8 ;
No
?- zaria(4, [(1,2),(2,3),(3,4)], [1], Moves).
Moves = 3 ;
No
?- zaria(3, [(1,2),(2,3)], [4,2,6], Moves).
No
```

Puzzle (0.25 βαθμοί)



Το n-puzzle είναι μια κλασσική σπαζοκεφαλιά που είναι γνωστή με διάφορα ονόματα και παραλλαγές. Το παιχνίδι ξεκινάει από έναν αρχικό σχηματισμό και ο παίκτης πρέπει να μετακινήσει κάθε φορά ένα από τα κουτάκια με τα νούμερα στο κουτάκι που είναι κενό έτσι ώστε να καταλήξει στο σχηματισμό όπου όλα τα κουτάκια με τα νούμερα βρίσκονται στη σειρά και το κενό έχει μετακινηθεί στη δεξιά κάτω γωνία. Η πρώτη εικόνα στα δεξιά είναι από την αντίστοιχη σελίδα[1] της Wikipedia και δείχνει τη λυμένη μορφή του 15-puzzle. Όπως αναφέρεται στην ίδια ιστοσελίδα δεν είναι πάντα δυνατό να καταλήξουμε σε λύση του n-puzzle από κάθε αρχικό σχηματισμό και η δεύτερη εικόνα δείχνει έναν τέτοιο σχηματισμό (όπου τα τετράγωνα 14 και 15 δεν είναι στη σωστή τους διάταξη).

Η άσκηση θα επικεντρώσει σε μια ακόμα μικρότερη μορφή του puzzle, το 8-puzzle. Επίσης, για να μην υπάρχει αμφισημία θεωρείστε ότι, αντί να μετακινείται κάποιο τετράγωνο με αριθμό, αυτό που μετακινείται κάθε φορά πάνω (**`u`**), κάτω (**`d`**), δεξιά (**`r`**) ή αριστερά (**`l`**) είναι το τετράγωνο με το κενό (το συμβολίζουμε με ένα **`x`**) όπως δείχνεται στο παρακάτω παράδειγμα.

```
1 2 3    d    1 2 3    r    1 2 3    r    1 2 3
x 5 6    →    4 5 6    →    4 5 6    →    4 5 6
4 7 8         x 7 8         7 x 8         7 8 x
```

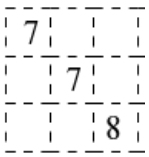
1 http://en.wikipedia.org/wiki/Fifteen_puzzle

Γράψτε ένα πρόγραμμα σε Prolog το οποίο να ορίζει ένα κατηγόρημα **puzzle/2** το οποίο να επιτυγχάνει βρίσκοντας τη λύση του 8-puzzle με τις λιγότερες δυνατές κινήσεις ξεκινώντας από τον αρχικό σχηματισμό (αν το puzzle έχει λύση) και να αποτυγχάνει αν το puzzle δε μπορεί να επιλυθεί.

Μερικά παραδείγματα συμπεριφοράς του προγράμματός σας δείχνονται παρακάτω (η πρώτη ερώτηση δείχνει την είσοδο που αντιστοιχεί στο σχηματισμό που χρησιμοποιήσαμε παραπάνω ως παράδειγμα).

```
?- puzzle([1,2,3,x,5,6,4,7,8], Moves).
Moves = [d,r,r] ;
No
?- puzzle([1,2,3,4,5,6,7,8,x], Moves).
Moves = [] ;
No
?- puzzle([7,6,8,x,2,3,4,1,5], Moves).
No
```

Ουκ όντους (0.25+0.25 βαθμοί)

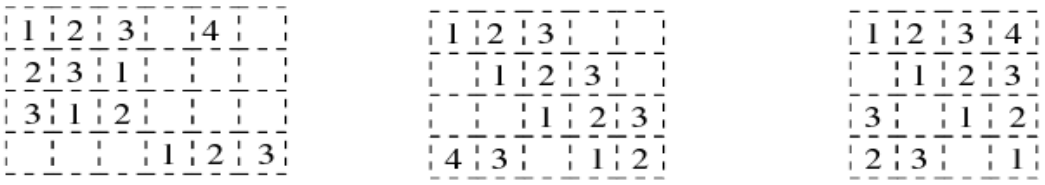


Το *ουκ όντους* είναι το εξής puzzle. Δοθείσης μιας λίστας από ακεραίους, πιθανώς με επαναλήψεις, τοποθετείστε τους σε ένα παραλληλόγραμμο έτσι ώστε:

1. Καμία γραμμή ή στήλη του παραλληλόγραμμου να μην περιέχει κάποιον αριθμό περισσότερες από μία φορά και
2. Το συνολικό εμβαδό του παραλληλόγραμμου να είναι το ελάχιστο δυνατό.

Η πρώτη απαίτηση ικανοποιείται πολύ εύκολα: με το να βάλουμε τον κάθε αριθμό σε διαφορετική γραμμή και στήλη όπως φαίνεται και στο πρώτο σχήμα στο πλάι για τη λίστα αριθμών **[7,7,8]**. Όμως με τη διάταξη αυτή το εμβαδό είναι 9 ενώ υπάρχει (ελάχιστη) λύση όπου το παραλληλόγραμμο έχει εμβαδό 4, όπως φαίνεται και δίπλα.

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 λύνουν το puzzle και να επιστρέφουν τις δύο διαστάσεις του παραλληλόγραμμου. Στην παρακάτω εικόνα δείχνουμε τρία πιθανά παραλληλόγραμμα για τη λίστα των αριθμών **[1,1,1,1,2,2,2,2,3,3,3,3,4]**. Το δεξιότερο είναι αυτό με το ελάχιστο εμβαδό για τους συγκεκριμένους αριθμούς. Όπως φαίνεται και στο σχήμα, δεν είναι ανάγκη οι αριθμοί να είναι συνεχόμενοι.



Μερικά από τα αποτελέσματα του προγράμματος σε Prolog φαίνονται παρακάτω:

```
?- ukodus([7,7,8], R, C).
R = 2
C = 2 ;
No
?- ukodus([1,1,1,1,2,2,2,2,3,3,3,3,4], R, C).
R = 4
C = 4 ;
No
```

```
?- ukodus([1,3,2], R, C).
R = 1
C = 3 ;
No
```

Όπως φαίνεται και στο τρίτο παράδειγμα, όταν οι διαστάσεις του παραλληλόγραμμου διαφέρουν μεταξύ τους επιλέγουμε τη μικρότερη από αυτές για το δεύτερο όρισμα του κατηγορήματος **ukodus/3** και τη μεγαλύτερη για το τρίτο.

Το πρόγραμμά σας σε ML πρέπει να έχει τη συμπεριφορά εισόδου/εξόδου που φαίνεται παρακάτω:

```
- ukodus [1,3,2];
val it = (1,3) : 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 στην επόμενη γραμμή):

```
> java Ukodus.class
είσοδος
     3
     1 3 2
έξοδος
     1
     3
```

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.