ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Α' Ομάδα Ασκήσεων "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2021-22

Άσκηση 1

Μία ακολουθία από σύμβολα μπορεί να κωδικοποιηθεί σύμφωνα με τη λογική του *τρέχοντος μήκους* (run length encoding) με τον εξής τρόπο. Κάθε μέγιστη υπο-ακολουθία από το ίδιο σύμβολο *S* μήκους *N* (> 1) κωδικοποιείται σαν ένα ζευγάρι *(S,N)*. Υπο-ακολουθίες μήκους 1 δεν κωδικοποιούνται, αλλά απλά παραμένει στο αποτέλεσμα το σύμβολο *S*. Για παράδειγμα, η ακολουθία "`a, a, a, b, b, c, d, d, d, d, e`" κωδικοποιείται σαν "`(a,3), (b,2), c, (d,4), e`". Ορίστε ένα κατηγόρημα Prolog `decode_rl/2`, το οποίο όταν καλείται με πρώτο όρισμα μία λίστα που παριστάνει μία κωδικοποιημένη ακολουθία συμβόλων, σύμφωνα με τα παραπάνω, να την αποκωδικοποιεί και να επιστρέφει στο δεύτερο όρισμα το αποτέλεσμα. Τα σύμβολα μπορεί να είναι όροι Prolog, όχι όμως μεταβλητές. Μπορούν να είναι όμως δομές που περιέχουν μεταβλητές. Κάποια παραδείγματα εκτέλεσης:

```
?- decode_rl([(a,3),(b,2),c,(d,4),e], L).
L = [a,a,a,b,b,c,d,d,d,d,e]

?- decode_rl([(f(5,a),7)], L).
L = [f(5,a),f(5,a),f(5,a),f(5,a),f(5,a),f(5,a),f(5,a)]

?- decode_rl([g(X),(h(Y),3),k(Z),(m(W),4),n(U)], L).
L = [g(X),h(Y),h(Y),h(Y),k(Z),m(W),m(W),m(W),m(W),n(U)]
```

Στη συνέχεια, υλοποιήστε σε Prolog και την κωδικοποίηση τρέχοντος μήκους, ορίζοντας ένα κατηγόρημα `encode_rl/2`, το οποίο να επιτελεί την αντίστροφη λειτουργία από αυτήν του `decode_rl/2`. Δηλαδή να παίρνει σαν πρώτο όρισμα μία λίστα συμβόλων (όρων Prolog πλην μεταβλητών) και να επιστρέφει στο δεύτερο όρισμα την κωδικοποιημένη εκδοχή της. Παραδείγματα:

```
?- encode_rl([a,a,a,b,b,c,d,d,d,d,e], L).
L = [(a,3),(b,2),c,(d,4),e]

?- encode_rl([f(5,a),f(5,a),f(5,a),f(5,a),
              f(5,a),f(5,a),f(5,a)], L).
L = [(f(5,a),7)]

?- encode_rl([g(X),h(Y),h(Y),h(Y),k(Z),
              m(W),m(W),m(W),m(W),n(U)], L).
L = [g(X),(h(Y),3),k(Z),(m(W),4),n(U)]
```

Τι απάντηση θα δοθεί, με βάση τον ορισμό που δώσατε, στην εξής ερώτηση;

```
?- encode_rl([p(3),p(X),q(X),q(Y),q(4)], L).
```

Εξηγήστε γιατί παίρνετε τη συγκεκριμένη απάντηση μέσω σχολίων στο αρχείο που θα παραδώσετε.

Παραδοτέο για την άσκηση είναι **<u>ένα</u> πηγαίο αρχείο Prolog** με όνομα **runlen.pl**, στο οποίο θα περιέχονται οι ορισμοί τόσο του `decode_rl/2` όσο και του `encode_rl/2`.

<u>Άσκηση 2</u>

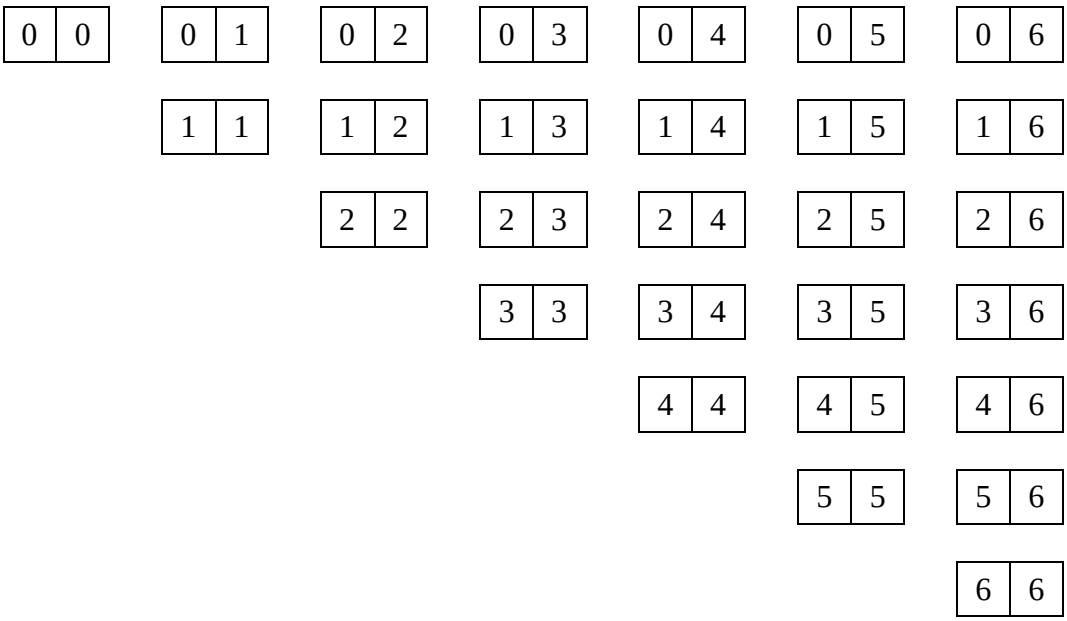
Έστω ότι έχετε στη διάθεσή σας έναν επεξεργαστή που διαθέτει N καταχωρητές R_1, R_2, ..., R_N οι οποίοι έχουν δακτυλιοειδή διευθέτηση. Αυτό σημαίνει ότι μπορεί κάποιος να μεταφέρει τα περιεχόμενα του καταχωρητή R_i στον καταχωρητή R_{i+1}, για $1 \leq i < N$, και του R_N στον R_1 με τις εντολές `move`(i), για $1 \leq i < N$, και `move`(N), αντίστοιχα. Επίσης, μπορεί να αντιμεταθέσει τα περιεχόμενα των καταχωρητών R_i και R_j με την εντολή `swap`(i,j), όπου $i < j$. Έστω, τώρα, ότι σας δίνονται τα αρχικά περιεχόμενα των N καταχωρητών, καθώς και τα επιθυμητά τελικά περιεχόμενα. Το ζητούμενο είναι να βρεθεί η μικρότερη αλληλουχία από τις εντολές `move` και `swap` που πρέπει να εκτελεσθούν για να επιτευχθεί ο ζητούμενος μετασχηματισμός. Συγκεκριμένα, ορίστε το κατηγόρημα `codegen/3`, έτσι ώστε όταν αυτό καλείται με πρώτο όρισμα τη λίστα των αρχικών περιεχομένων των καταχωρητών και με δεύτερο όρισμα τη λίστα των τελικών περιεχομένων, να επιστρέφει στο τρίτο όρισμα τη λίστα των απαραίτητων (ελάχιστων) εντολών που απαιτούνται για το μετασχηματισμό. Σημειώστε ότι είναι δυνατόν στην αναπαράσταση των περιεχομένων των καταχωρητών να έχουμε, τόσο στην αρχική όσο και στην τελική κατάσταση, το σύμβολο `*`, που σημαίνει, για μεν την αρχική κατάσταση "δεν ξέρω τι περιέχεται στον καταχωρητή", για δε την τελική κατάσταση "δεν με ενδιαφέρει τι περιέχεται στον καταχωρητή". Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- codegen([a,b,c,d],[a,d,a,b],L).
L = [move(2),move(1),swap(3,4),swap(2,3)]

?- codegen([a,*,c],[c,a,*],L).
L = [move(1),move(3)]

?- codegen([a,b,c],[a,a,*],L).
L = [move(1)]

?- codegen([a,b,c,d,e,f],[f,f,b,e,a,e],L).
L = [move(2),swap(4,6),move(5),swap(4,5),swap(1,5),move(1)]
```

Παραδοτέο για την άσκηση είναι **<u>ένα</u> πηγαίο αρχείο Prolog** με όνομα **codegen.pl**.

Άσκηση 3

Έστω ότι έχουμε στη διάθεσή μας τις παρακάτω 28 πλάκες ντόμινο:



Γράψτε ένα πρόγραμμα Prolog που να μας πληροφορεί πώς πρέπει να τις τοποθετήσουμε σ' ένα πλαίσιο 7x8, έτσι ώστε η τελική διάταξη να είναι η εξής:

3	1	2	6	6	1	2	2
3	4	1	5	3	0	3	6
5	6	6	1	2	4	5	0
5	6	4	1	3	3	0	0
6	1	0	6	3	2	4	0
4	1	5	2	4	3	5	5
4	1	0	2	4	5	2	0

Κωδικοποιήστε τα δεδομένα του προβλήματος ως εξής:

```
dominos([(0,0),(0,1),(0,2),(0,3),(0,4),(0,5),(0,6),
                (1,1),(1,2),(1,3),(1,4),(1,5),(1,6),
                      (2,2),(2,3),(2,4),(2,5),(2,6),
                            (3,3),(3,4),(3,5),(3,6),
                                  (4,4),(4,5),(4,6),
                                        (5,5),(5,6),
                                              (6,6)]).
```

```
frame([[3,1,2,6,6,1,2,2],
       [3,4,1,5,3,0,3,6],
       [5,6,6,1,2,4,5,0],
       [5,6,4,1,3,3,0,0],
       [6,1,0,6,3,2,4,0],
       [4,1,5,2,4,3,5,5],
       [4,1,0,2,4,5,2,0]]).
```

Ως προς τον τρόπο εμφάνισης του αποτελέσματος, ορίστε ένα κατηγόρημα `put_dominos/0`, το οποίο όταν καλείται, να εκτυπώνει το ζητούμενο. Για παράδειγμα:

```
?- put_dominos.
3-1 2 6 6 1 2-2
    | | | |
3-4 1 5 3 0 3 6
            | |
5 6-6 1 2-4 5 0
|     |
5 6 4 1 3 3-0 0
  | |   |     |
6 1 0 6 3 2 4 0
|     |   | |
4 1-5 2 4 3 5 5
        |     |
4-1 0-2 4 5-2 0
```

Φροντίστε, η υλοποίησή σας να μην είναι δεσμευτική για τα συγκεκριμένα δεδομένα, αλλά να μπορεί να εφαρμοστεί και για άλλο σύνολο από ντόμινο ή/και πλαίσιο.

Σημειώστε ότι ένα ντόμινο μπορεί να τοποθετηθεί στο πλαίσιο με οποιοδήποτε από τους τέσσερις δυνατούς τρόπους. Δηλαδή το ντόμινο

2	5

μπορεί να τοποθετηθεί με κάποιον από τους εξής τρόπους:

2	5

2
5

5	2

5
2

Παραδοτέο για την άσκηση είναι **<u>ένα</u> πηγαίο αρχείο Prolog** με όνομα **domino.pl**.