ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΑΘΗΜΑ: ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
ΑΚΑΔ. ΕΤΟΣ: 2015-16
ΔΙΔΑΣΚΩΝ: Χ.ΝΟΜΙΚΟΣ

4η Σειρά Εργαστηριακών Ασκήσεων

Οι απαντήσεις θα πρέπει να υποβληθούν με **turnin**, το αργότερο μέχρι την **Παρασκευή 27 Μαΐου 2016**, ώρα **20:00**.

- Για τη συγγραφή των προγραμμάτων θα πρέπει να χρησιμοποιήσετε το αρχείο πρότυπο Lab4.pro (που υπάρχει στην ιστοσελίδα του μαθήματος), στο οποίο για κάθε κατηγόρημα που ζητείτε να ορίσετε στις παρακάτω ασκήσεις, υπάρχει ένας κανόνας ο οποίος το ορίζει έτσι ώστε να επιστρέφει πάντα την απάντηση no. Για να απαντήσετε στις ασκήσεις αντικαταστήστε τους παραπάνω κανόνες με ένα κατάλληλο σύνολο προτάσεων που να ορίζει το κάθε κατηγόρημα. **Δεν θα πρέπει να τροποποιήσετε το όνομα κανενός κατηγορήματος ούτε το πλήθος των ορισμάτων του**.

- Μπορείτε να χρησιμοποιήσετε όσα βοηθητικά κατηγορήματα θέλετε, τα οποία θα χρησιμοποιούνται για τον ορισμό των κατηγορημάτων που σας ζητείται να υλοποιήσετε. Σε καμία περίπτωση δεν θα πρέπει να προσθέσετε άλλα ορίσματα στα κατηγορήματα που σας ζητούνται.

- Μετά το τέλος της εκφώνησης κάθε άσκησης δίνονται παραδείγματα ερωτήσεων με τις αντίστοιχες αναμενόμενες απαντήσεις, που μπορείτε να χρησιμοποιήσετε για έλεγχο της ορθότητας των προγραμμάτων σας.

- Ο έλεγχος της ορθότητας των απαντήσεων θα γίνει με ημι-αυτόματο τρόπο. Σε καμία περίπτωση δεν θα πρέπει ο βαθμολογητής να χρειάζεται να κάνει παρεμβάσεις στο αρχείο που θα υποβάλετε. Συνεπώς θα πρέπει να λάβετε υπόψη τα παρακάτω:

    1. Κάθε ένα από τα κατηγορήματα που σας ζητείται να υλοποιήσετε θα πρέπει να έχει το συγκεκριμένο όνομα και το συγκεκριμένο πλήθος ορισμάτων που περιγράφεται στην εκφώνηση της αντίστοιχης άσκησης και που υπάρχει στο αρχείο πρότυπο Lab4.pro. **Αν σε κάποια άσκηση το όνομα ή το πλήθος των ορισμάτων δεν συμφωνεί με αυτόν που δίνεται στην εκφώνηση, η άσκηση δεν θα βαθμολογηθεί**.
    2. Το αρχείο που θα παραδώσετε δεν θα πρέπει να περιέχει συντακτικά λάθη. Αν υπάρχουν τμήματα κώδικα που περιέχουν συντακτικά λάθη, τότε θα πρέπει να τα διορθώσετε ή να τα αφαιρέσετε πριν από την παράδοση. **Αν το αρχείο που θα υποβάλετε περιέχει συντακτικά λάθη, τότε ολόκληρη η εργαστηριακή άσκηση θα μηδενιστεί**.

3. Οι ερωτήσεις που δίνονται στο τέλος κάθε άσκησης θα πρέπει να επιστρέφουν απάντηση. Αν κάποιες από τις επιστρεφόμενες απαντήσεις δεν είναι σωστές, αυτό θα ληφθεί υπόψη στη βαθμολογία, ωστόσο η άσκηση θα βαθμολογηθεί κανονικά. **Αν ωστόσο κάποια από τις παραπάνω ερώτησεις δεν επιστρέφει απάντηση, (π.χ. προκαλείται υπερχείλιση στοίβας, ατέρμονος υπολογισμός ή κάποιο σφάλμα χρόνου εκτέλεσης) τότε ο βαθμός για την υλοποίηση του αντίστοιχου κατηγορήματος θα είναι μηδέν**.
4. Κατα τη διόρθωση των ασκήσεων οι βαθμολογητές δεν θα κάνουν χρησιμοποιήσουν ερωτήσεις που εμπεριέχουν τα βοηθητικά κατηγορήματα τα οποία ενδεχομένως θα έχετε ορίσει. Η χρήση των βοηθητικών κατηγορημάτων θα πρέπει να γίνεται μέσα από τα κατηγορήματα που σας ζητείται να υλοποιήσετε.

- Για υποβολή με turnin γράψτε (υποθέτοντας ότι έχετε γράψει το πρόγραμμα στο αρχείο Lab4.pro):

turnin Prolog-2@myy401 Lab4.pro

Ασκηση 1.

Γράψτε ένα πρόγραμμα σε Prolog το οποίο με δεδομένους δύο πραγματικούς αριθμούς A και X θα υπολογίζει τον ελάχιστο θετικό ακέραιο N για τον οποίο η τιμή του αθροίσματος $\sum_{i=1}^{N}(1-\frac{A}{i})$ γίνεται μεγαλύτερη ή ίση του X. Συγκεκριμένα, ορίστε ένα κατηγόρημα p(X,A,N) το οποίο θα αληθεύει το N είναι ο ελάχιστος θετικός α-κέραιος τέτοιος ώστε $\sum_{i=1}^{N}(1-\frac{A}{i}) \geq X$.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- p(3,0,N).
N = 3

| ?- p(1,1,N).
N = 3

| ?- p(3,1,N).
N = 6

| ?- p(100,0,N).
N = 100

| ?- p(100,1,N).
N = 106

| ?- p(100,2,N).
N = 111

| ?- p(100,10,N).
N = 157

| ?- p(50,-1,N).
N = 46

| ?- p(-30,200,N).
N = 1556

| ?- p(-30,0,N).
N = 1

| ?- p(1000,3,N).
N = 1023

| ?- p(10000,3,N).
N = 10030
```

Ασκηση 2.
Ορίστε ένα κατηγόρημα sumOf2Cubes(N) σε Prolog το οποίο θα αληθεύει αν το N είναι το άθροισμα των κύβων δύο θετικών ακεραίων.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- sumOf2Cubes(2).
yes

| ?- sumOf2Cubes(5).
no

| ?- sumOf2Cubes(16).
yes

| ?- sumOf2Cubes(25).
no

| ?- sumOf2Cubes(129).
no

| ?- sumOf2Cubes(351).
yes

| ?- sumOf2Cubes(1027).
yes

| ?- sumOf2Cubes(91234).
no

| ?- sumOf2Cubes(659834).
yes
```

Ασκηση 3.

Με δεδομένες δύο λίστες L, S και ένα στοιχείο X θέλουμε να κατασκευάσουμε τη λίστα που προκύπτει από την L αντικαθιστώντας, για κάθε i, την i-οστή εμφάνιση του X με το i-στό στοιχείο της S. Αν το πλήθος εμφανίσεων της X στην L είναι μεγαλύτερο από το μήκος της S, τότε οι πλεονάζουσες εμφανίσεις του X θα πρέπει να διαγράφονται.

Γράψτε ένα πρόγραμμα σε Prolog το οποίο θα πραγματοποιεί την παραπάνω διαδικασία. Συγκεκριμένα ορίστε ένα κατηγόρημα replace(L,X,S,R), το οποίο θα αληθεύει αν η R προκύπτει από την L αν i-οστή εμφάνιση του X αντικατασταθεί με το i-στό στοιχείο της S ή διαγραφεί αν η S περιέχει λιγότερα από i στοιχεία.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- replace([1,2,3,4,5],4,[0],R).
R = [1,2,3,0,5]

| ?- replace([1,2,3,4,5],0,[],R).
R = [1,2,3,4,5]

| ?- replace([],a,[a,b,c,c,b],R).
R = []

| ?- replace([a,b,c,d,e,f],k,[x,y,z],R).
R = [a,b,c,d,e,f]

| ?- replace([1,2,3,2,4,3,3,1],2,[5,0,7],R).
R = [1,5,3,0,4,3,3,1]

| ?- replace([a,b,a,a,c,d,a,e,a],a,[0,1],R).
R = [0,b,1,c,d,e]

| ?- replace([a,b,a,a,c,b,d],b,[5,4,3,2,1,0],R).
R = [a,5,a,a,c,4,d]
```

Ασκηση 4.
Σε μία ψηφοφορία για την ανάδειξη του καλύτερου ποδοσφαιριστή της χρονιάς, ψηφίζουν n δημοσιογράφοι και νικητής είναι όποιος συγκεντρώσει περισσότερες από $n/2$ ψήφους. Αν κανένας υποψήφιος δεν συγκεντρώσει περισσότερες από $n/2$ ψήφους τότε δεν υπάρχει νικητής.

Γράψτε ένα πρόγραμμα σε Prolog το οποίο με δεδομένη μία λίστα με τις ψήφους των δημοσιογράφων θα βρίσκει το νικητή της ψηφοφορίας, εφόσον υπάρχει. Συγκεκριμένα, ορίστε ένα κατηγόρημα majority(L,X), το οποίο θα αληθεύει αν το X εμφανίζεται σε περισσότερες θέσεις από όσο είναι το μισό του μήκους της λίστας L.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- majority([],X).
no
| ?- majority(['Messi'],X).
X = 'Messi'
| ?- majority(['Ronaldo', 'Messi'],X).
no

| ?- majority(['Ronaldo', 'Ronaldo', 'Messi', 'Messi', 'Ronaldo',
'Ronaldo', 'Ronaldo', 'Pogba', 'Buffon'],X).
X = 'Ronaldo'

| ?- majority(['Ronaldo', 'Messi', 'Messi', 'Vidal', 'Messi',
'Messi', 'Pogba', 'Buffon'],X).
no

| ?- majority(['Messi', 'Ronaldo', 'Messi', 'Pogba', 'Pogba',
'Pogba', 'Pogba'],X).
X = 'Pogba'

| ?- majority(['Vidal', 'Vidal', 'Vidal', 'Vidal', 'Messi'],X).
X = 'Vidal'

| ?- majority(['Messi', 'Messi', 'Messi', 'Ronaldo', 'Pogba', 'Ronaldo',
'Ronaldo', 'Buffon', 'Messi', 'Messi', 'Messi'],X).
X = 'Messi'

| ?- majority(['Buffon', 'Messi', 'Buffon', 'Ronaldo', 'Buffon',
'Ronaldo', 'Buffon', 'Messi', 'Buffon'],X).
X = 'Buffon'

| ?- majority(['Buffon', 'Messi', 'Pogba', 'Ronaldo', 'Vidal', 'Ronaldo',
'Buffon', 'Messi', 'Buffon', 'Pogba'],X).
no
```

Ασκηση 5.
Σε ένα παιχνίδι που μοιάζει με το ντόμινο, δίνεται ένα πλήθος από λίστες και το ζητούμενο είναι να τοποθετηθούν οι λίστες αυτές η μία μετά την άλλη έτσι ώστε το τελευταίο στοιχείο κάθε λίστας να είναι το πρώτο στοιχείο της επόμενης.

Ορίστε ένα κατηγόρημα domino(L) σε Prolog , το οποίο θα αληθεύει αν η L είναι μία μη κενή λίστα από λίστες οι οποίες μπορούν να τοποθετηθούν στη σειρά με τον περιορισμό που περιγράφεται παραπάνω.

Για έλεγχο χρησιμοποιήστε τις παρακάτω ερωτήσεις:

```
| ?- domino([]).
no

| ?- domino([[1,2,3]]).
yes

| ?- domino([[1,2,3],[3,2,1]]).
yes

| ?- domino([[1,2,1],[1,3,3,1],[1,4,4,1],[1,1],[1],[1,0,1],[1,9,0,1]]).
yes

| ?- domino([[1,2,3],[1],[3]]).
yes

| ?- domino([[1,2,3],[],[3,2,1]]).
no

| ?- domino([[3,3,2,1,4],[1,3,5,7,2],[4,3,3,2,5],[2,4,3,1,3]]).
yes

| ?- domino([[1,2],[1,3],[1],[2,1],[2,3],[2],[3,1],[3,2],[3]]).
yes

| ?- domino([[1,4],[2,6],[3,7],[4,2],[5,3],[6,9],[7,1],[8,5]]).
yes

| ?- domino([[1,2],[2,3],[2,8],[3,4],[3,5],[4,1],[5,1],[8,4],[8,5]]).
no

| ?- domino([[0,3],[1,6],[2,8],[3,7],[4,0],[5,9],[6,4],[7,1],[8,5],[9,2]]).
no
```