ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΑΘΗΜΑ: ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
ΑΚΑΔ. ΕΤΟΣ: 2016-17
ΔΙΔΑΣΚΩΝ: Χ.ΝΟΜΙΚΟΣ

4η Σειρά Εργαστηριακών Ασκήσεων

Οι εργαστηριακές ασκήσεις είναι ατομικές. Οι απαντήσεις θα πρέπει να υποβληθούν με **turnin**, το αργότερο μέχρι την **Παρασκευή 26 Μαΐου 2017**.

Οι ώρες οι οποίες έχουν δεσμευτεί για το εργαστήριο του μαθήματος είναι την Παρασκευή 5-7μμ. Η παρουσία στο εργαστήριο τις παραπάνω ώρες δεν είναι υποχρεωτική. Μπορείτε να έρχεστε στο εργαστήριο τις ώρες αυτές για όποια βοήθεια χρειάζεστε σχετικά με την εκπόνηση των εργαστηριακών ασκήσεων και γενικότερα τον προγραμματισμό στη γλώσσα Prolog, καθώς και για την επίλυση προβλημάτων που παρουσιάζονται κατά τη συγγραφή των προγραμμάτων στο πλαίσιο των εργαστηριακών ασκήσεων. **Σημειώνεται ότι σε καμία περίπτωση δεν θα παρέχεται σχετική βοήθεια μέσω e-mail.**

- 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.

- 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.

- 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.

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

turnin Prolog-2@myy401 Lab4.pro

Ασκηση 1.
Μπορούμε να αναπαραστήσουμε κλάσματα στην Prolog χρησιμοποιώντας όρους της μορφής A/B, όπου A και B είναι θετικοί ακέραιοι αριθμοί. Γράψτε ένα πρόγραμμα σε Prolog για την υλοποίηση της πρόσθεσης δύο κλασμάτων. Συγκεκριμένα, ορίστε ένα κατηγόρημα fracSum(X,Y,Z) το οποίο θα αληθεύει αν ισχύουν τα παρακάτω:

- X,Y,Z είναι κλάσματα τέτοια ώστε X + Y = Z.
- Το Z είναι ένα απλοποιημένο κλάσμα, δηλαδή ο αριθμητής και ο παρονομαστής του δεν έχουν κανέναν κοινό διαιρέτη μεγαλύτερο του 1.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- fracSum(5/9,2/9,Z).
Z = 7/9
| ?- fracSum(1/2,1/3,Z).
Z = 5/6
| ?- fracSum(3/1,5/1,Z).
Z = 8/1
| ?- fracSum(5/9,1/9,Z).
Z = 2/3
| ?- fracSum(3/8,5/8,Z).
Z = 1/1
| ?- fracSum(3/2,9/4,Z).
Z = 15/4
| ?- fracSum(1/4,1/12,Z).
Z = 1/3
| ?- fracSum(5/6,7/6,Z).
Z = 2/1
| ?- fracSum(3/5,3/20,Z).
Z = 3/4
| ?- fracSum(2/7,1/3,Z).
Z = 13/21
```

Ασκηση 2.
Γράψτε ένα πρόγραμμα σε Prolog για τον υπολογισμό του αθροίσματος όλων των θετικών ακεραίων που διαιρούν έναν δεδομένο θετικό ακέραιο N. Συγκεκριμένα, ορίστε ένα κατηγόρημα sumd(N,S) το οποίο θα αληθεύει αν το N είναι θετικός ακέραιος και η τιμή της S ισούται με την με το άθροισμα όλων των των θετικών ακεραίων που διαιρούν το N.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- sumd(1,N).
N = 1
| ?- sumd(2,N).
N = 3
| ?- sumd(4,N).
N = 7
| ?- sumd(8,N).
N = 15
| ?- sumd(10,N).
N = 18
| ?- sumd(17,N).
N = 18
| ?- sumd(24,N).
N = 60
| ?- sumd(100,N).
N = 217
| ?- sumd(101,N).
N = 102
| ?- sumd(999,N).
N = 1520
```

Ασκηση 3.
Έστω μία λίστα αριθμών L. Ένα στοιχείο της λίστας ονομάζεται:

- α-κορυφή αν είναι μεγαλύτερο από το προηγούμενο του στοιχείο στη λίστα L και δεν είναι μικρότερο από το επόμενο του στοιχείο στη λίστα L.
- α-πυθμένας αν είναι μικρότερο από το προηγούμενο του στοιχείο στη λίστα L και δεν είναι μεγαλύτερο από το επόμενο του στοιχείο στη λίστα L.

Ονομάζουμε α-περίληψη της L τη λίστα που περιέχει το πρώτο και το τελευταίο στοιχείο της L, καθώς και όλα τα στοιχεία της που είναι είτε α-κορυφές είτε α-πυθμένες, με τη ίδια σειρά εμφάνισης όπως στην L. Αν η L είναι κενή τότε και η α-περίληψή της είναι κενή.

Γράψτε ένα πρόγραμμα σε Prolog το οποίο να υπολογίζει την α-περίληψη μίας δεδομένης λίστας. Συγκεκριμένα ορίστε ένα κατηγόρημα alpha(L,S), το οποίο θα αληθεύει αν L και S είναι λίστες και η S είναι η α-περίληψη της L.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- alpha([],L).
L = []
| ?- alpha([1],L).
L = [1]
| ?- alpha([1,2],L).
L = [1,2]
| ?- alpha([1,2,3,4,5],L).
L = [1,5]
| ?- alpha([5,4,3,2,1],L).
L = [5,1]
| ?- alpha([1,2,3,3,5],L).
L = [1,3,5]
| ?- alpha([1,3,2,4,5],L).
L = [1,3,2,5]
| ?- alpha([4,1,1,2,3],L).
L = [4,1,3]
| ?- alpha([1,2,3,5,4],L).
L = [1,5,4]
| ?- alpha([1,3,3,3,4,5,5],L).
L = [1,3,5,5]
yes
```

Ασκηση 4.
Μπορούμε να προσθέσουμε στο τέλος μίας λίστας L επαναληπτικά το στοιχείο nil, μέχρι να προκύψει λίστα το μήκος της οποίας είναι δύναμη του 2. Ονομάζουμε τη λίστα που προκύπτει με αυτή τη διαδικασία 2^n-επέκταση της L. Αν το μήκος της L είναι δύναμη του 2, τότε η 2^n-επέκτασή της είναι η ίδια η L.

Γράψτε ένα πρόγραμμα σε Prolog το οποίο να υπολογίζει τη 2^n-επέκταση μίας δεδομένης λίστας. Συγκεκριμένα ορίστε ένα κατηγόρημα expand(L,E), το οποίο θα αληθεύει αν L και E είναι λίστες και η E είναι η 2^n-επέκταση της L.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
| ?- expand([],E).
E = [nil]
| ?- expand([1],E).
E = [1]
| ?- expand([1,2],E).
E = [1,2]
| ?- expand([1,2,3],E).
E = [1,2,3,nil]
| ?- expand([1,2,3,4,5],E).
E = [1,2,3,4,5,nil,nil,nil]
| ?- expand([1,2,3,4,5,6],E).
E = [1,2,3,4,5,6,nil,nil]
| ?- expand([1,2,3,4,5,6,7],E).
E = [1,2,3,4,5,6,7,nil]
| ?- expand([a,b,c,d,e,f,g,h],E).
E = [a,b,c,d,e,f,g,h]
| ?- expand([0,0,0,0,0,0,0,0,0],E).
E = [0,0,0,0,0,0,0,0,0,nil,nil,nil,nil,nil,nil,nil]
| ?- expand([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],E).
E = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,nil]
```