Ασκήσεις Φυλλάδιο 5

Cut, μεταβλητή κλήση και δημιουργία/αποδόμηση όρων

(ή αλλιώς "The power of backtracking, logic and negation combined!").

1. Έστω ο ακόλουθος ορισμός για το μέγιστο δύο αριθμών:
max1(X,Y,X):- X>=Y.
max1(X,Y,Y):- X=<Y.

max2(X,Y,X):- X>=Y, !.
max2(X,Y,Y).

Ποια η διαφορά των δύο κατηγορημάτων? Ποιο πρόβλημα μπορεί να προκύψει από το κατηγόρημα **max2/3**? (red cut).

2. Χρησιμοποιώντας τον τελεστή ! να ορίσετε ένα κατηγόρημα **set_diff/3** το οποίο υλοποιεί αφαίρεση συνόλων στα δύο πρώτα του κατηγορήματα τα οποία είναι λίστες. Για παράδειγμα:

?- set_diff([1,2,3,4],[1,2],L).
L = [3, 4]
Yes
?- set_diff([1,2,3,4],[1,2,5,6],L).
L = [3, 4]
Yes
?- set_diff([1,2,3,4],[1,2,5,3,6],L).
L = [4]
Yes

3. Να ορίσετε το κατηγόρημα **lunion(List1,List2,List3)** χρησιμοποιώντας ! που επιτυγχάνει όταν η λίστα **List3** είναι η συνένωση των στοιχείων της **List1** που δεν υπάρχουν στην **List2,** με την **List2.** Για παράδειγμα:

?- lunion([a, b, c], [d, e, f, a], L).
L = [b, c, d, e, f, a]

?- lunion([a, b], [d, e, f, b, a], L).
L = [d, e, f, b, a]

?- lunion([a, b, c], [d, e, f, b], L).
L = [a, c, d, e, f, b]

4. Χρησιμοποιώντας το κατηγόρημα **not/1** να ορίσετε το κατηγόρημα **max_list(Max,List)**, που επιτυγχάνει όταν το **Max** είναι το μέγιστο στοιχείο της λίστας **List**.

5. Να ορίσετε ένα κατηγόρημα **unique_element(X,List**), το οποίο επιτυγχάνει όταν το **X** είναι στοιχείο της λίστας **List** και εμφανίζεται μόνο μια φορά. Να χρησιμοποιήσετε το κατηγόρημα **not/ 1**. Για παράδειγμα

?- unique_element(X, [a, b, c, c, b, d]).
X = a ;
X = d ;
No

?- unique_element(X, [a, a, b, c, c, b]).
No

6. Να ορίσετε ένα κατηγόρημα **proper_set(List)**, το οποίο επιτυγχάνει όταν η λίστα **List** περιέχει μόνο μοναδικά στοιχεία, είναι δηλαδή σύνολο. Να χρησιμοποιήσετε τον ορισμό του **unique_element/2** που δώσατε στην άσκηση 5 και το **not/1**. Για παράδειγμα:

?- proper_set([1, 2, 1, 3]).
No
?- proper_set([1, 2, 3]).
Yes
?- proper_set([a, b, c, d]).
Yes
?- proper_set([a, b, c, d, d]).
No

7. Να ορίσετε το κατηγόρημα **map(Operation,List,Results)** το οποίο δεδομένου ενός ονόματος κατηγορήματος τάξης 2 **Operation,** και μιας λίστας **List,** επιστρέφει τη λίστα **Results** που περιέχει τα αποτελέσματα της εφαρμογής του κατηγορήματος στη δοθείσα λίστα:

?- map(double, [1, 2, 3, 4], L).
L = [2, 4, 6, 8]
Yes
?- map(square, [1, 2, 3, 4], L).
L = [1, 4, 9, 16]
Yes

Οι ορισμοί των **double(X,Y)** και **square(X,Y)** θα δοθούν από εσάς.

?- double(3,X)
X = 6
?- square(3,X)
X = 9

8. Να ορίσετε ένα κατηγόρημα **reduce(Operation, List, Result)**, το οποίο αν το **Operation** είναι το όνομα ενός κατηγορήματος με τάξη 3 το οποίο υλοποιεί μια πράξη ανάμεσα σε δύο ακέραιους και η λίστα **List** είναι μια λίστα ακεραίων, τότε η μεταβλητή **Result** ενοποιείται με το αποτέλεσμα της διαδοχικής εφαρμογής του **Operation** σε ζεύγη της λίστας **List.** Για παράδειγμα

?- reduce(max, [2, 34, 2, 3, 45], L).
L = 45
?- reduce(min, [2, 34, 2, 3, 45], L).
L = 2
?- reduce(plus, [2, 3, 4], L).
L = 9
?- reduce(times, [2, 3, 4], L).
L = 24

Τα **max/3, min/3, plus/3, times/3** κλπ είναι υλοποιημένα από την EcliPSe Prolog.

9. Να ορίσετε ένα κατηγόρημα **valid_queries/1** το οποίο όταν παίρνει σαν όρισμα μια ερώτηση (στόχο) της Prolog, η οποία περιέχει ένα αριθμό ελεύθερων μεταβλητών, τυπώνει στην οθόνη όλες τις πλήρως ορισμένες έγκυρες ερωτήσεις (δηλ αυτές που αποτιμώνται σε true), Για παράδειγμα:

?- valid_queries(member(X,[1,2,3,4])).
member(1, [1, 2, 3, 4])
member(2, [1, 2, 3, 4])
member(3, [1, 2, 3, 4])
member(4, [1, 2, 3, 4])
No

Τι θα συμβεί αν υπάρχουν άπειρες απαντήσεις, όπως για παράδειγμα στην ερώτηση
append(L1,L2,L) ;

10. Έστω το Prolog κατηγόρημα **seperate_lists(List,Lets,Nums)** το οποίο δοθείσας μιας λίστας ακεραίων και χαρακτήρων (ατόμων) **List**, πετυχαίνει όταν στην λίστα **Lets** είναι όλοι οι χαρακτήρες (άτομα) της **List** και στην λίστα **Nums**, όλοι οι αριθμοί. Για παράδειγμα:

?- seperate_lists([1, 2, a, b, c, 3, 4, 5, f], Lets, Nums).
Lets = [a, b, c, f]
Nums = [1, 2, 3, 4, 5]
Yes (0.00s cpu)
?- seperate_lists([1, 2], Lets, Nums).
Lets = []
Nums = [1, 2]
Yes (0.00s cpu)
?- seperate_lists([a, b], Lets, Nums).
Lets = [a, b]
Nums = []
Yes (0.00s cpu)

Να δώσετε τον **αναδρομικό ορισμό** του κατηγορήματος κάνοντας όσο το δυνατό λιγότερους ελέγχους,
(Σημ: Το κατηγόρημα **number(N)** πετυχαίνει όταν το N είναι αριθμός).

11. Να ορίσετε το κατηγόρημα **max_min_eval(List,Result)** το οποίο δέχεται μια παράσταση

ακεραίων και πράξεων **min**/**max** σε μορφή λίστας ([2,max,3,min,1] == 2 max 3 min 1), και ενοποιεί το **Result**, με το αποτέλεσμα της παράστασης. Θεωρείστε τα min και max ως τελεστές, οι οποίοι έχουν ίδια προτεραιότητα και είναι αριστερά προσεταιριστικοί, δηλαδή x max y min z == (x max y) min z.
Για παράδειγμα [2,max,3,min,1] == 2 max 3 min 1 == (2 max 3) min 1 == 3 min 1 == 1.
Σε περίπτωση που η παράσταση δεν είναι συντακτικά ορθή, τότε το κατηγόρημα αποτυγχάνει.
Για παράδειγμα:
?- max_min_eval([2, min, 3], Result).
Result = 2
Yes
?- max_min_eval([2, min, 3, max, 10], Result).
Result = 10
Yes
?- max_min_eval([2, min, 3, min, 10], Result).
Result = 2
Yes
?- max_min_eval([2, min, 3, min, 10, max, 30], Result).
Result = 30
Yes
?- max_min_eval([2, min, 3, min, 10, max], Result).
No
?- max_min_eval([2, foo, 3], Result).
No