ΕΡΓΑΣΤΗΡΙΟ VI

ΑΝΑΔΡΟΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ ΛΙΣΤΕΣ

ΜΕΡΟΣ Ι

ΛΙΣΤΕΣ

Η λίστα είναι μια πολύ χρήσιμη δομή δεδομένων για προγραμματισμό σε Prolog. *Μια λίστα είναι ουσιαστικά ένας δυναμικός μονοδιάστατος πίνακας με στοιχεία : αριθμούς, σύμβολα, σύνθετους όρους ή και άλλες λίστες.*

Η κενή λίστα συμβολίζεται με : [].

Η λίστα είναι μια δομή με δύο όρους :

- Την κεφαλή (head) που είναι το πρώτο στοιχείο της λίστας και
- Την ουρά (tail) που είναι το υπόλοιπο τμήμα της λίστας

Μία λίστα που περιέχει στοιχεία απεικονίζεται για παράδειγμα ως εξής : [physics,8,maths,9] όπου τα στοιχεία χωρίζονται μεταξύ τους με κόμμα.

Μια λίστα μπορεί να ενοποιηθεί με μια μεταβλητή π.χ.

X= [physics,8,maths,9]

Τις περισσότερες φορές θέλουμε να χειριστούμε τα στοιχεία της λίστας ένα-ένα. Σ' αυτή την περίπτωση μπορούμε να ενοποιήσουμε μια μεταβλητή με το πρώτο στοιχείο της λίστας (κεφαλή) και μια άλλη μεταβλητή με τα υπόλοιπα στοιχεία της λίστας (ουρά). Έτσι η προσπάθεια ταυτοποίησης :

[X|Y] = [physics,8,maths,9] *μας δίνει* **X=physics** *και* **Y=[8,maths,9]**

Η μεταβλητή X μπορεί να έχει οποιαδήποτε μορφή. Η Y πρέπει να είναι αποκλειστικά λίστα. Ο τελεστής " | " αποτελεί ειδική αναπαράσταση, που δείχνει άμεσα την κεφαλή και την ουρά μιας λίστας.

ΜΕΡΟΣ ΙΙ

ΟΡΙΣΜΟΣ ΣΧΕΣΕΩΝ ΛΙΣΤΩΝ

Η σχέση **member(X,Y)** είναι αληθής, όταν το X είναι στοιχείο της λίστας Y.

```
member(X,[X|_]).
member(X,[_|Z]):- member(X,Z).
```

Το νόημα του παραπάνω προγράμματος είναι : «*Το X είναι μέλος μιας λίστας αν είναι η κεφαλή της λίστας ή αν είναι μέλος της ουράς της λίστας*»

Η σχέση **append(L1,L2,L3)** είναι αληθής, όταν η L3 είναι η λίστα που προκύπτει από τη συνένωση των λιστών L1 και L2.

```
append([],L,L).
append([X|L1],L2,[X|L3]):-append(L1,L2,L3).
```

Η δομή του παραπάνω προγράμματος είναι παρόμοια με αυτή του προγράμματος για τον ορισμό της σχέσης plus.

Η σχέση **reverse(X,Y)** ορίζεται σαν συνάρτηση της *append*. Είναι αληθής, όταν η λίστα Y είναι η αντίστροφη της λίστας X

```
reverse([],[]).
reverse([X|Xs],Ys):-reverse(Xs,Rs), append(Rs, [X], Ys).
```

Το νόημα του παραπάνω προγράμματος είναι : «*Η αντίστροφη της κενής λίστας είναι η κενή λίστα. Η αντίστροφη μη κενής λίστας μπορεί να παραχθεί αντιστρέφοντας την ουρά της και προσκολλώντας στο τέλος της αντιστραμμένης ουράς την κεφαλή της αρχικής λίστας*».
Η σχέση **sumlist(L,Sum)** υπολογίζει το Sum αθροίζοντας τα στοιχεία της λίστας L.

```
sumlist([],0).
sumlist([X|L],Sum) :- sumlist(L,TempSum), Sum is TempSum+X.
```

Η σχέση **last(X,Y)** επιστρέφει σαν Y το τελευταίο στοιχείο της λίστας X.

```
last([X],X).
last([_|L1],X) :- last(L1,X).
```

Το νόημα του παραπάνω προγράμματος είναι : «*Για να είναι ένα στοιχείο, το τελευταίο μιας λίστας θα πρέπει είτε να είναι το μοναδικό στοιχείο της λίστας, είτε να είναι το τελευταίο στοιχείο της ουράς της*.»

Η σχέση **delete(X,L1,L2)** , όπου η λίστα L2 προκύπτει από την L1 αφού έχουν διαγραφεί όλες οι εμφανίσεις του X.

```
delete(_X,[],[]).
delete(X,[X|L],L2) :- delete(X,L,L2).
delete(X,[X1|T1],[X1|T2]) :- X\=X1, delete(X,T1,T2).
```

Η σχέση **suffix(X,Y)** είναι αληθής, όταν το X είναι το τελευταίο τμήμα (επίθεμα) της λίστας Y.

```
suffix(L,L).
suffix(L,[H|T]) :- suffix(L,T).
```

Ξαναορίζουμε τη σχέση member χρησιμοποιώντας τη σχέση append:

```
member1(X,L) :- append(L1, X|L2,L).
```

Η σχέση **add(X,L1,L2)** είναι αληθής όταν η λίστα L2 προκύπτει από την L1 με την προσθήκη του X.

```
add(X,L,[X|L]).
```

Χρησιμοποιήστε την trace για να ελέγξετε την λειτουργία των παραπάνω σχέσεων αφού ολοκληρώσετε τις παρακάτω ασκήσεις.

ΑΣΚΗΣΕΙΣ

Χρησιμοποιώντας το πρόγραμμα exercise6.pl που θα βρείτε έτοιμο στην διαδρομή: http://iiwm.teikav.edu.gr/digital_lessons/ στο μάθημα Λογική και Λογικός Προγραμματισμός – Ενότητα 6 απαντήστε στα παρακάτω ερωτήματα :

Α. Εστω η σχέση:

```
sumlist([],0).
sumlist([X|L],Sum) :- sumlist(L,TempSum), Sum is TempSum+X.
```

Διατυπώστε την ερώτηση: **sumlist([1,2,3], X)** χρησιμοποιώντας την trace για να κατανοήσετε τη λειτουργία του παραπάνω κανόνα.

Β. Ποια θα πρέπει να είναι η ερώτηση που θα κάνετε στην Prolog για να σας απαντήσει αν το x είναι μέλος της λίστας [1,x,45,α,X]; Τι θα σας απαντήσει η Prolog; Γιατί;

Γ. Εστω η ερώτηση **member([2006, hellas], X).** Τι εμφανίζει σαν αποτέλεσμα και γιατί;

Δ. Ποια είναι τη σημασία του παρακάτω κανόνα; Διατυπώστε δύο ερωτήσεις: μία που να επιστρέφει true και μία που να επιστρέφει false (χρησιμοποιήστε την εντολή trace για την κατανόηση του κανόνα).

```
subset([],_).
subset([X |R], L):-member(X,L),subset(R,L).
```

Ε. Ποια θα πρέπει να είναι η ερώτηση που θα κάνετε στην Prolog για να σας εμφανίσει ποια λίστα πρέπει να συνενώσετε με την λίστα [physics,6,maths,11] για να σας δώσουν τη λίστα [chemistry,7,greek,10,physics,6,maths,11].

ΣΤ. Ποια θα πρέπει να είναι η ερώτηση που θα κάνετε στην Prolog για να σας εμφανίσει ποια είναι η αντίστροφη της λίστας [chemistry,7,greek,10,physics,6,maths,11].

Ζ. Έστω η σχέση:

```
last1(X,[X]).
last1(X,[_|L1]):-last1(X,L1).
```

Χρησιμοποιώντας την εντολή trace, να διατυπώσετε την ερώτηση
last1(X, [2,3]). Τι ακριβώς κάνει η last1(X,Y).;

Η. Δημιουργήστε την σχέση front(X,Y) η οποία είναι αληθής, όταν το X είναι το πρώτο στοιχείο της λίστας Y.

ΜΕΡΟΣ ΙΙΙ

ΛΙΣΤΕΣ ΩΣ ΣΥΝΘΕΤΟΙ ΟΡΟΙ

Οι λίστες στην Prolog μπορεί να θεωρηθεί ότι είναι σύνθετοι όροι τάξης 2, όπου το πρώτο όρισμα είναι η κεφαλή της λίστας και το δεύτερο η ουρά της (σε μορφή λίστας). Ως συναρτησιακό σύμβολο χρησιμοποιείται η τελεία "**.**". Έτσι ισχύουν οι παρακάτω ισοδυναμίες :

[α,β] = .(α,.(β,[]))

[1] = .(1,[])

[3,7,16] = .(3,.(7,.(16,[])))

[[2],g] = .(.(2,[]),.(g,[]))

Ουσιαστικά, δηλαδή, η αναπαράσταση με αγκύλες είναι για την ευκολία του προγραμματιστή και την αναγνωσιμότητα των προγραμμάτων.

Ασκήσεις για εξάσκηση πάνω στην ύλη του εργαστηρίου

1. Να ορισθεί το κατηγόρημα not_member(X,L), το οποίο θα αληθεύει όταν το στοιχείο X δεν περιέχεται στη λίστα L.

2. Να ορισθεί το κατηγόρημα length_list /2, το οποίο έχει ως πρώτο όρισμα τη λίστα και ως δεύτερο θα επιστρέφει το μήκος της.

3. Να ορισθεί το κατηγόρημα adjacent(X, Y, L) έτσι ώστε να επιστρέφει true αν τα X, Y είναι γειτονικά στη λίστα L, και false σε οποιαδήποτε άλλη περίπτωση.