ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

6° Εξάμηνο

Περισσότερα για λίστες
(Λίστες με στοιχεία Λίστες)

Δημοσθένης Σταμάτης
http:/www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

Διαχωρισμός λίστας ακεραίων σε 2 λίστες:

Μία με τους θετικούς και αρνητικούς και μία με τους αρνητικούς

```
posneg([ ], [ ], [ ]).
```

Διαχωρισμός λίστας ακεραίων σε 2 λίστες:

Μία με τους θετικούς και αρνητικούς και μία με τους αρνητικούς

```
posneg([ ], [ ], [ ]).

posneg([H|T],        ,    ) :-
        H > 0,
        posneg(T, TP, TN).
```


Διαχωρισμός λίστας ακεραίων σε 2 λίστες:

Μία με τους θετικούς και αρνητικούς και μία με τους αρνητικούς

```
posneg([ ], [ ], [ ]).

posneg([H|T], [H|TP], TN) :-
        H > 0,
        posneg(T, TP, TN).
```

Διαχωρισμός λίστας ακεραίων σε 2 λίστες:

Μία με τους θετικούς και αρνητικούς και μία με τους αρνητικούς

```
posneg([ ], [ ], [ ]).

posneg([H|T], [H|TP], TN) :-
        H > 0,
        posneg(T, TP, TN).

posneg([H|T], TP, [H|TN]) :-
        H < 0,
        posneg(T, TP, TN).
```


Διαχωρισμός λίστας ακεραίων σε 2 λίστες:

Μία με τους θετικούς και αρνητικούς και μία με τους αρνητικούς

```
posneg([ ], [ ], [ ]).

posneg([H|T], [H|TP], TN) :-
        H > 0,
        posneg(T, TP, TN).

posneg([H|T], TP, [H|TN]) :-
        H < 0,
        posneg(T, TP, TN).

posneg([H|T], TP, TN) :-
        H = 0,
        posneg(T, TP, TN).
```

?- posneg([13,-51,-11,29], LP,LN).
LP=[13,29] LN=[-51,-11]

?- posneg([1, 3, 0, 21], LP,LN).
LP=[1,3,21] LN=[ ]

Διαχωρισμός λίστας ακεραίων σε 2 λίστες:

Άμεση ενοποίηση των ορισμάτων

```
posneg([ ], [ ], [ ]).

posneg([H|T], [H|TP], TN) :-
        H > 0,
        posneg(T, TP, TN).

posneg([H|T], TP, [H|TN]) :-
        H < 0,
        posneg(T, TP, TN).

posneg([H|T], TP, TN) :-
        H = 0,
        posneg(T, TP, TN).
```

Λοξοκοιτώντας τη Java!

```
posneg(L, L1, L2) :-
    split(L,H,T),
    H > 0,
    posneg(T, TP, TN),
    insertfirst(H,TP, L1),
    L2  = TN.

split([H|T], H, T).
insertfirst(H, T, [H|T]).
```


Ένωση Συνόλων

(αναπαράστασή τους με λίστες)

```
sets_union([H|T], L1, [H|L2]) :-
        not(member(H, L1)),
        sets_union(T, L1, L2).
```

Ένωση Συνόλων
(αναπαράστασή τους με λίστες)

```
sets_union([H|T], L1, [H|L2]) :-
        not(member(H, L1)),
        sets_union(T, L1, L2).

sets_union([H|T], L1, L2) :-
        member(H, L1),
        sets_union(T, L1, L2).
```


Ένωση Συνόλων
(αναπαράστασή τους με λίστες)

```
sets_union([ ], L, L).

sets_union([H|T], L1, [H|L2]) :-
        not(member(H, L1)),
        sets_union(T, L1, L2).

sets_union([H|T], L1, L2) :-
        member(H, L1),
        sets_union(T, L1, L2).
```

```
?- sets_union([a, b, s, d, k], [v, w, b, k, x], L).
   L = [a, s, d, v, w, b, k, x]
```

Δημιουργία ενός Συνόλου από τα στοιχεία 2 λιστών

```
union([ ], L, L).

union([H|T], L1, [H|L2]) :-
        union(T, L1, L2),
        not(member(H, L2)).

union([H|T], L1, L2) :-
        union(T, L1, L2),
        member(H, L2).
```

```
?- sets_union([a,b,s,d,s,d,k], [v,w,b,k,x], L).
   L = [a, s, d, s, d, v, w, b, k, x] .
?- union([a,b,s,d,s,d,k], [v,w,b,k,x], L).
   L = [a, s, d, v, w, b, k, x]
```


Μία λίστα αποτελείται από λίστες. Να βρεθεί στο άθροισμα των μηκών τους (= πόσα απλά στοιχεία έχει)

```
total_length([ ], 0).

total_length([H|T], SL) :-
        total_length(T, SLT),
        length(H, LH),
        SL is SLT + LH.
```

```
?- total_length([[2,3], [1,s,d,a], [3]], X).
   X = 7.
?- total_length([[ ], [1,3,2], [a,d,s]], X).
   X=6
```

Μία λίστα έχει στοιχεία που μπορεί να είναι είτε σταθερές είτε (αναδρομικά) λίστες. Να γίνει επίπεδη !

```
flat([ ], [ ]).
```


Μία λίστα έχει στοιχεία που μπορεί να είναι είτε σταθερές είτε (αναδρομικά) λίστες. Να γίνει επίπεδη !

```
flat([ ], [ ]).

flat([H|T], [H|LT].) :-
        atomic(H),
        flat(T, LT).
```

Μία λίστα έχει στοιχεία που μπορεί να είναι είτε σταθερές είτε (αναδρομικά) λίστες. Να γίνει επίπεδη !

```
flat([ ], [ ]).

flat([H|T], [H|LT].) :-
        atomic(H),
        flat(T, LT).

flat([H|T], L) :-
        not(atomic(H)),
        flat(T, LT),
        flat(H, LH),
        append(LH, LT, L).
```

```
?- flat([ [a, e], [ [ [b], c] ] ], L1).
   L2 = [a, e, b, c]
?- flat([a, [ [b,c], [ [d,e], f] ], g], L2).
   L2 = [a, b, c, d, e, f, g]
```


Μία λίστα έχει στοιχεία που μπορεί να είναι είτε σταθερές είτε (αναδρομικά) λίστες. Να γίνει επίπεδη !

```
flat([ ], [ ]).

flat([[ ]|T], L) :-
        flat(T, L).

flat([H|T], [H|LT].) :-
        atomic(H),
        flat(T, LT).

flat([H|T], L) :-
        not(atomic(H)),
        flat(T, LT),
        flat(H, LH),
        append(LH, LT, L).
```

```
/*
?- flat([a, [ [b,c], [ [d,e], f] ], [ ], g], L2).
   L2 = [a, b, c, d, e, f, [ ], g]
*/
?- flat([a, [ [b,c], [ [d,e], f] ], [ ], g], L2).
   L2 = [a, b, c, d, e, f, g]
```

Μία λίστα αποτελείται από λίστες δεδομένων.
Να βρεθεί εάν κάποιο δεδομένο ανήκει σ' αυτή.

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

memberlist(X, [_|T])  :-
        memberlist(X, T).
```

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

```
?- memberlist(X, [ [b,c], [d,e], [f,2,a,3] ]).
   true
?- memberlist(X, [ [b,c], [d,e], [f,2,a,3] ]).
   X = b ;
   X = c ;
   X = d  . . . κ.ο.κ.
```


Έξυπνη χρήση της append: Έλεγχος Ορθογραφίας

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

```
?- append(M1, M2, [t,e,x,t]).
M1 = [ ],          M2 = [t, e, x, t] ;
M1 = [t],          M2 = [e, x, t] ;
M1 = [t, e],       M2 = [x, t] ;
M1 = [t, e, x],    M2 = [t] ;
M1 = [t, e, x, t], M2 = [ ]
```

Έξυπνη χρήση της append: Έλεγχος Ορθογραφίας

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

word([t, e, x, t] ).
```

```
?- word(W), append(M1, M2, W).
M1 = [ ],          M2 = [t, e, x, t] ;
M1 = [t],          M2 = [e, x, t] ;
M1 = [t, e],       M2 = [x, t] ;
M1 = [t, e, x],    M2 = [t] ;
M1 = [t, e, x, t], M2 = [ ]
```


Έξυπνη χρήση της append: Έλεγχος Ορθογραφίας

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

word([t, e, x, t] ).
```

```
?- ?- word(L), append(M1,[x|M2],L).
L = [t, e, x, t],
M1 = [t, e],
M2 = [t]
```

. . . Περισσότερα στην επόμενη άσκηση πράξης