

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ, ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Λογικός Προγραμματισμός (ΤΛΕΣ): 2^η Ατομική Εργασία

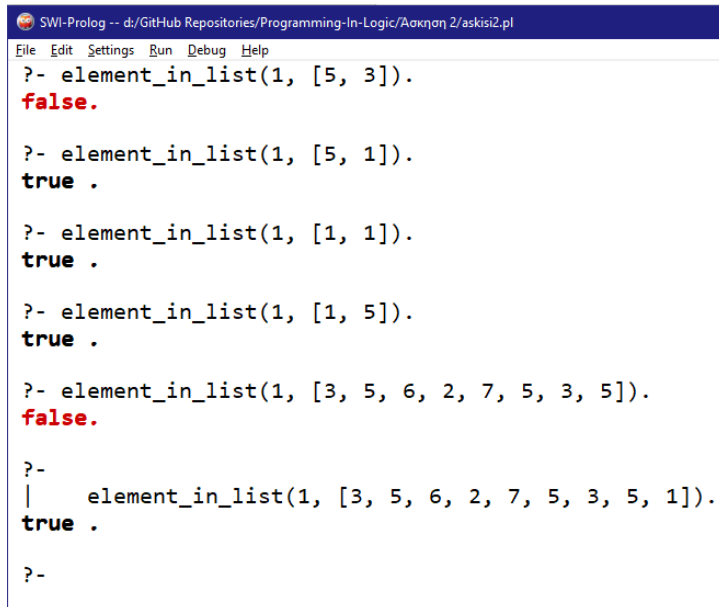
ΓΕΩΡΓΙΟΣ ΣΕΪΜΕΝΗΣ – Π19204

Ως εργασία ζητήθηκε να φτιαχτεί μία συνάρτηση `common_list/2`, η οποία θα επέστρεφε `true`, αν δύο λίστες έχουν τουλάχιστον ένα κοινό στοιχείο μεταξύ τους. Στο αρχείο `askisi2.pl` βρίσκεται η προγραμματιστική προσέγγιση για την λύση της δεύτερης Άσκησης.

Στις γραμμές 2 και 3 βρίσκεται ο αναδρομικός ορισμός `element_in_list/2`, για το αν ένα στοιχείο υπάρχει μέσα σε μία λίστα. Το πρώτο όρισμα είναι το στοιχείο που ψάχνουμε. Το δεύτερο όρισμα είναι η λίστα στην οποία ψάχνουμε το στοιχείο. Οπότε, για να το κοιτάξουμε αναδρομικά αυτό, πρώτα ορίζουμε (στην γραμμή 2) τι ισχύει, όταν μια λίστα έχει μονάχα ένα στοιχείο.

```
2 element_in_list(X, [X]).
3 element_in_list(X, [Y | T]) :- X == Y; element_in_list(X, T).
```

Στην γραμμή 3, κοιτάμε αναδρομικά τον κανόνα και παράλληλα κοιτάμε αν η κεφαλή της λίστας είναι το στοιχείο που ψάχνουμε. Με αυτόν τον τρόπο, επιτυγχάνουμε το να ελέγξουμε όλα τα στοιχεία της λίστας και να φτιάξουμε τη συνάρτηση που θα μας βοηθήσει αρκετά στο να βρούμε αν δύο λίστες έχουν τουλάχιστον ένα κοινό στοιχείο.



```
SWI-Prolog -- d:/GitHub Repositories/Programming-In-Logic/Άσκηση 2/askisi2.pl
File Edit Settings Run Debug Help
?- element_in_list(1, [5, 3]).
false.

?- element_in_list(1, [5, 1]).
true.

?- element_in_list(1, [1, 1]).
true.

?- element_in_list(1, [1, 5]).
true.

?- element_in_list(1, [3, 5, 6, 2, 7, 5, 3, 5]).
false.

?-
| element_in_list(1, [3, 5, 6, 2, 7, 5, 3, 5, 1]).
true.

?-
```

Στην παραπάνω εικόνα, φαίνεται πως, πράγματι, η συνάρτηση που φτιάξαμε, δουλεύει για όλες τις λίστες οποιουδήποτε μεγέθους. Οπότε, μπορούμε να την χρησιμοποιήσουμε για την `common_list/2`, η οποία θα δέχεται δύο λίστες ως ορίσματα.

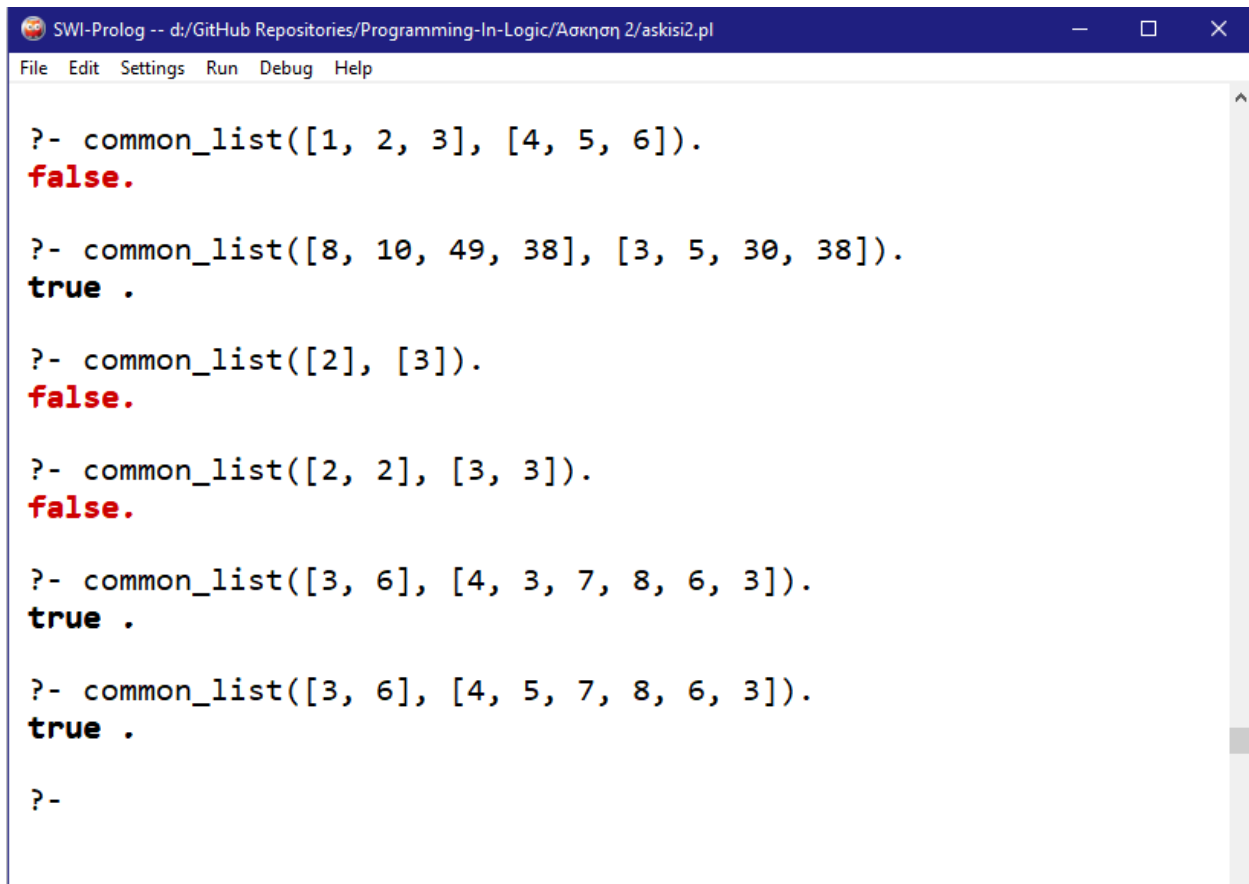
Με τον ίδιο τρόπο, δηλαδή αναδρομικά, προσπαθούμε να προσεγγίσουμε το αν δυο λίστες έχουν το ίδιο στοιχείο. Στην γραμμή 6, θα ορίσουμε, πρώτα, τι ισχύει για λίστες με μονάχα ένα στοιχείο. Ύστερα, μπορούμε να λύσουμε αναδρομικά το θέμα, έχοντας στα χέρια μας την προηγούμενη συνάρτηση.

```
6 common_list([X], [X]).
7 common_list([X | T1], [Y | T2]) :- element_in_list(X, [Y | T2]); element_in_list(Y, [X | T1]); common_list(T1, T2).
```

Στην γραμμή 7 χωρίζουμε τις δύο λίστες σε υπολίστες, έχοντας, πρώτα, πάρει τις κεφαλές τους. Οπότε, κοιτάμε αν η κεφαλή της πρώτης λίστας ανήκει στην δεύτερη λίστα ή αν η κεφαλή της δεύτερης λίστας ανήκει στην πρώτη λίστα. Σε οποιοδήποτε από τα δύο σενάρια η συνάρτηση αυτή θα επιστρέψει true, αν όντως υπάρχουν. Το μόνο που απομένει, είναι να δούμε τις δύο υπολίστες τις οποίες θα τις κοιτάξουμε αναδρομικά.

Σημαντική Σημείωση: Η συνάρτηση δουλεύει καλύτερα αν και οι δύο λίστες έχουν το ίδιο μέγεθος.

Ακολουθούν χαρακτηριστικά παραδείγματα που αποδεικνύουν τη λειτουργικότητα της συναρτήσεως.



```
SWI-Prolog -- d:/GitHub Repositories/Programming-In-Logic/Άσκηση 2/askisi2.pl
File Edit Settings Run Debug Help

?- common_list([1, 2, 3], [4, 5, 6]).
false.

?- common_list([8, 10, 49, 38], [3, 5, 30, 38]).
true .

?- common_list([2], [3]).
false.

?- common_list([2, 2], [3, 3]).
false.

?- common_list([3, 6], [4, 3, 7, 8, 6, 3]).
true .

?- common_list([3, 6], [4, 5, 7, 8, 6, 3]).
true .

?-
```

ΤΕΛΟΣ ΠΑΡΟΥΣΙΑΣΗΣ.

~ ΓΕΩΡΓΙΟΣ ΣΕΪΜΕΝΗΣ – Π19204