ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΕΥΤΕΡΗ ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΠΕΙΡΑΙΑΣ 2020

Η Βάση Γνώσης της Εργασίας

```
1   prefix([], _).
2
3   % helper predicate: first list is prefix of the second list
4   prefix([X | Tail_X], [X | Tail_Y]) :-
5       prefix(Tail_X, Tail_Y).
6
7   % check if the Tail of each list are the same (recursion)
8   included_list([X | Tail_X], [X | Tail_Y]) :-
9       prefix(Tail_X, Tail_Y).
10
11  % check if X is part of list Y
12  included_list(X, [_ | Tail_Y]) :-
13      included_list(X, Tail_Y).
```

Υλοποίηση & Παραδείγματα Ορθής Εκτέλεσης

Η λογική για την επίλυση του προβλήματος είναι ότι πρώτα θα χρειαστούμε έναν βοηθόκατηγόρημα (**prefix/2**) για να ελέγξουμε ότι η πρώτη λίστα είναι πρόθεμα της δεύτερης. Η «χειρότερη» περίπτωση είναι ότι εξαντλούμε τη λίστα-πρόθεμα που σημαίνει ότι δεν βρέθηκε η λίστα X εντός της Y. Η περίπτωση που επιτυγχάνει η επαγωγή είναι ότι τα τρέχοντα στοιχεία ταιριάζουν και το υπόλοιπο και των δύο λιστών είναι ένα πρόθεμα.

Για παράδειγμα έχουμε: included_list([2, 3], [1, 2, 3, 4, 5]).

- Αποτυγχάνει η ενοποίηση (unification) στη γραμμή 8 καθώς τα Heads των δύο λιστών είναι διαφορετικά.
- Επόμενο βήμα της Prolog, είναι να κοιτάξει στη γραμμή 12 όπου η ενοποίηση επιτυγχάνει με X = [2, 3] και Tail_Y = [2, 3, 4, 5].
- Στη συνέχεια, γίνεται έλεγχος αν το X ανήκει στο Tail_Y, δηλαδή το X = 2 υπάρχει μέσα στο Tail_Y = [3, 4, 5] κάτι το οποίο δεν ισχύει.
- Προσπαθεί ξανά από την αρχή και επιτυγχάνει η ενοποίηση αυτή τη φορά, στη γραμμή 8 με X = 2, Tail_X = [3], Tail_Y = [3, 4, 5] και γίνεται έλεγχος με την βοήθεια της **prefix/2,** δηλαδή αν το Tail_X αποτελεί πρόθεμα του Tail_Y, κάτι το οποίο επιτυγχάνει.
- Προσπαθεί ξανά από την αρχή και πλέον γίνεται ενοποίηση με το prefix/2 στη γραμμή 4 με X =3, Tail_X = [ ] και Tail_Y = [4, 5]. Γίνεται ο έλεγχος και επιτυγχάνει.

Γενικά, ψάχνουμε να βρούμε μια διαδοχική υπολίστα (στο παράδειγμα μας 2, 3) που είναι ίση με τη λίστα-πρόθεμα. Αν τα στοιχεία ταιριάξουν τότε η λίστα-πρόθεμα βρέθηκε άρα είναι και υπολίστα της λίστας Y. Σε κάθε άλλη περίπτωση αφαιρούμε το στοιχείο με το οποίο ελέγχαμε και με αναδρομή προσπαθούμε ξανά παίρνοντας το επόμενο από την λίστα X.

Ακολουθούν μερικά παραδείγματα επιτυχής εκτέλεσης:

```
% c:/Users/bmano/Desktop/Prolog files/City-Hall-Planner/2nd assignment/list_sublis
t_of_a_second_list.pl compiled 0.00 sec, 0 clauses
?- included_list([], [1,2,3,4,5]).
false.

?- included_list([1,2,3], [1,2,3,4,5]).
true.

?- included_list([2,3], [1,2,3,4,5]).
true.

?- included_list([1,2,3,4,5], [1,2,3,4,5]).
true.

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

?- included_list([X], [1,2,3,4,5]).
X = 1 ;
X = 2 ;
X = 3 ;
X = 4 ;
X = 5 ;
false.
```

Τόσο ο κώδικας Prolog της εργασίας όσο και στιγμιότυπα επιτυχής εκτέλεσης του προγράμματος θα είναι διαθέσιμα στο GitHub μέσω του παρακάτω συνδέσμου:

https://github.com/vagman/logical-programming