ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΕΥΤΕΡΗ ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΠΕΙΡΑΙΑΣ 2021

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΑΥΓΕΡΙΝΟΣ ΧΡΗΣΤΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: Π19020

ΕΚΦΩΝΗΣΗ ΕΡΓΑΣΙΑΣ

[Για φοιτητές με επώνυμο από Α έως Ε] Να γραφεί το κατηγόρημα precede_list(X, Y) που αληθεύει όταν η X λίστα προηγείται της Y λίστας, δηλ. αν η X είναι υπολίστα με κάποια από τα πρώτα από αριστερά στοιχεία της Y.

Παράδειγμα:

?-precede_list([1,2], [1, 2, 3]).
Yes
?-precede_list([1,3], [1, 2, 3]).
No
?-precede_list([1], [1, 2, 3]).
Yes

<u>ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ:</u>

```
1
2  precede_list([],[H|T]). /*The first list is empty. Accept.*/
3
4  precede_list([H|T1],[H|T2]):-
5      /*The heads are the same. Check for the tails.*/
6      precede_list(T1,T2).
```

Υλοποίηση & Παραδείγματα Ορθής Εκτέλεσης

Η λογική για την επίλυση του προβλήματος είναι η χρήση ενός αναδρομικού κανόνα – κατηγορήματος precede_list/2 που θα τερματίζει όταν ισχύσει ο πρώτος κανόνας (τερματικός κανόνας) στη γραμμή 2, δηλαδή όταν το πρώτο όρισμα είναι μια κενή λίστα και το δεύτερο μια μη κενή λίστα. Συγκεκριμένα όταν ο χρήστης δώσει τις δυο λίστες X,Y (όπου η X πρέπει να προηγείται της Y) πρώτα ελέγχεται ο πρώτος κανόνας και αν και οι δυο λίστες είναι μη κενές τότε αποτυγχάνει και ελέγχεται ο δεύτερος στη γραμμή 4.Με τη σειρά του ο δεύτερος κανόνας σπάει τις δυο λίστες σε HEAD και TAIL και ελέγχει αν έχουν την ιδιά κεφαλή, δηλαδή αν το HEAD της πρώτης λίστας(X) ενοποιείται με το HEAD της δεύτερης λίστας(Y).Αν λοιπόν έχουν κοινή κεφαλή αληθεύει ο δεύτερος κανόνας και θα καλέσει αναδρομικά ξανά την διαδικασία αυτή τη φορά με τις ουρές των δυο λιστών δηλαδή με τα υπόλοιπα στοιχεία κάθε λίστας έως ότου αδειάσει η λίστα X και η Y έχει ακόμα στοιχεια.Τοτε θα επαληθευτεί ο πρώτος κανόνας, η αναδρομική διαδικασία θα τελειώσει και θα έχουμε true. Αντίθετα αν οι δυο λίστες δεν έχουν κοινή κεφαλή θα αποτύχει και ο δεύτερος κανόνας και θα έχουμε false.

Ακολουθούν μερικά παραδείγματα εκτέλεσης:

```
?- precede_list([1,2], [1, 2, 3]).
true.

?- precede_list([1,3], [1, 2, 3]).
false.

?- precede_list([1], [1, 2, 3]).
true.

?- precede_list([], [1, 2, 3]).
true.

?- precede_list([1,2], [1, 2]).
false.
```

ΕΞΤΡΑ ΕΡΩΤΗΜΑΤΑ

IV)

```
1
2   common_list([H|T],L):-
3       member(H,L),!;
4       common_list(T,L).
```

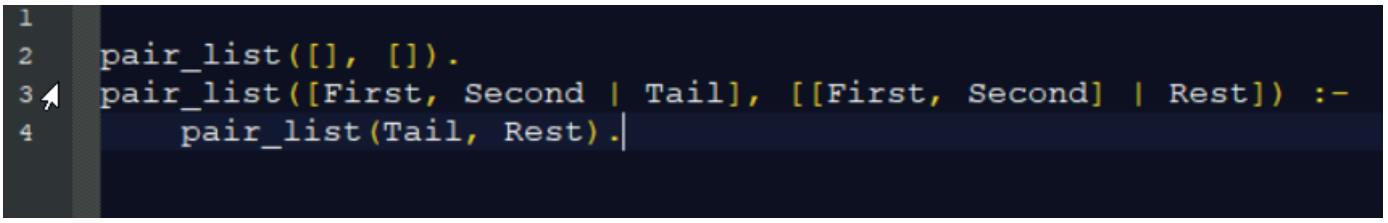
?- common_list([1,2, 3], [0, 3, 4]).
true.

?- common_list([1,2, 5], [0, 3, 4]).
false.

?- common_list([1,2, 4,0], [0, 3, 4]).
true.

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

V)



```
1
2   pair_list([], []).
3   pair_list([First, Second | Tail], [[First, Second] | Rest]) :-
4       pair_list(Tail, Rest).
```

?- pair_list([1,2, 3, 4, 5, 6], [[1, 2], [3, 4], [5, 6]]).
true.

?- pair_list([1,2, 3, 4, 5, 6], X).
X = [[1, 2], [3, 4], [5, 6]].