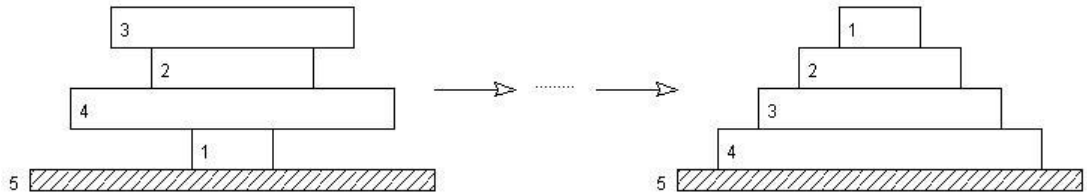
Θέμα 5, εργασία 1, 2021-2022: Προγραμματισμός σε Prolog [*10 μονάδες*]

Στην 1η γραπτή εργασία της ΠΛΗ31 το 2020-2021, ζητήθηκε από τους φοιτητές να επιλύσουν το πρόβλημα με τις πίτες (δείτε παρακάτω), τόσο μέσω αναζήτησης κατά βάθος (dfs), όσο και μέσω υλοποίησης της dfs σε Prolog.

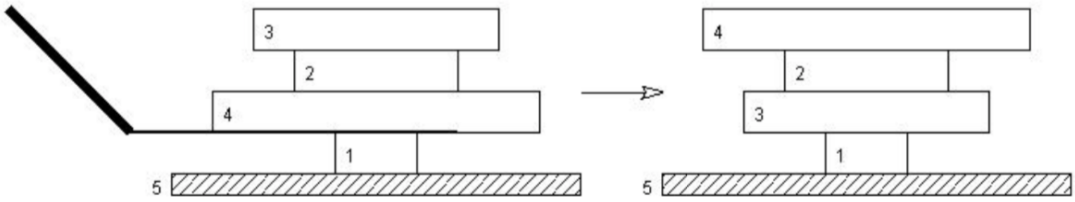
Εδώ καλείστε να επιλύσετε το ίδιο πρόβλημα (με τις πίτες) με Prolog, χρησιμοποιώντας μια διαφορετική προσέγγιση, γνωστής ως *generate-and-test*.

Το πρόβλημα με τις πίτες μπορεί να περιγραφεί ως εξής. Δίνονται N πίτες με διαμέτρους 1, 2, . . . , N, οι οποίες είναι τοποθετημένες σε μια στοίβα με τυχαία σειρά. Θέλουμε να ταξινομήσουμε τις πίτες στην στοίβα με αύξουσα σειρά (με την πίτα διαμέτρου 1 στην κορυφή της στοίβας) – βλ. Σχήμα 1 για παράδειγμα με N=4.



Σχήμα 1

Ο μοναδικός τρόπος για να αλλάξει η διάταξη της στοίβας είναι με την χρήση μια σπάτουλας. Τοποθετώντας την σπάτουλα κάτω από μια πίτα μπορεί κανείς να αναποδογυρίσει όλες της πίτες που βρίσκονται πάνω στην σπάτουλα και να τις τοποθετήσει στην στοίβα με ανεστραμμένη σειρά (βλ. Σχήμα 2).



Σχήμα 2

Συγκεκριμένα, αν η αρχική διάταξη στην στοίβα είναι (Σ_1, . . . , Σ_k, Σ_{k+1}, . . . Σ_N) και τοποθετήσουμε την σπάτουλα κάτω από την πίτα Σ_k, αναποδογυρίζοντας τις πίτες πάνω στην σπάτουλα παίρνουμε την στοίβα (Σ_k, . . . , Σ_1, Σ_{k+1}, . . . Σ_N).[1]

Επομένως για μια τυχαία αρχική διάταξη της στοίβας από πίτες, το ζητούμενο είναι να βρεθεί μια ακολουθία κινήσεων της σπάτουλας που ταξινομεί την στοίβα. Για παράδειγμα, μπορούμε να ταξινομήσουμε την στοίβα στα αριστερά του Σχήματος 1 με τις ακόλουθες κινήσεις:

1. Τοποθετούμε την σπάτουλα κάτω από την πίτα **2** και αναποδογυρίζουμε.
2. Τοποθετούμε την σπάτουλα κάτω από την πίτα **4** και αναποδογυρίζουμε.
3. Τοποθετούμε την σπάτουλα κάτω από την πίτα **1** και αναποδογυρίζουμε.

Η παραπάνω ακολουθία κινήσεων μπορεί να αναπαρασταθεί απλά με την ακολουθία αριθμών (2, 4, 1), με κάθε αριθμό να αντιστοιχεί στην πίτα κάτω από την οποία τοποθετείται κάθε φορά η σπάτουλα.

Στην επίλυση σας σε Prolog στο πρόβλημα με τις πίτες, θα πρέπει αναπαραστήσετε τόσο την στοίβα, όσο και μια ακολουθία κινήσεων της σπάτουλας, ως **λίστες**.

[1] Στην αναπαράσταση (Σ_1, . . ., Σ_N) της στοίβας, θεωρούμε πως το Σ_1 είναι στην κορυφή της στοίβας.

Για παράδειγμα η αριστερή στοίβα στο Σχήμα 1 θα πρέπει να αναπαρασταθεί με την λίστα [3, 2, 4, 1], και η ακολουθία κινήσεων που περιγράψαμε νωρίτερα για την ταξινόμηση αυτής της στοίβας, θα πρέπει να αναπαρασταθεί με την λίστα [2, 4, 1].

Επισημαίνεται πως η ακολουθία κινήσεων που ταξινομεί μια στοίβα είναι πιθανό να περιέχει τον ίδιο αριθμό περισσότερες από μια φορά.[2]

Ο κώδικάς σας θα πρέπει να ξεκινάει με τις παρακάτω γραμμές:

```
input([3,2,4,1]).
solution(S,Max) :-      input(L), sort(L,L1),
                        belongs(S,L,Max), turnSequence(S,L,L1).
```

Το κατηγόρημα input/1 ορίζει την στοίβα που θέλουμε να ταξινομήσουμε. Αρχικά θα χρησιμοποιήσουμε την στοίβα [3, 2, 4, 1] στα αριστερά του Σχήματος 1. Αργότερα, όταν ολοκληρώσετε τον κώδικα, μπορείτε να δοκιμάσετε διαφορετικές στοίβες στο input.

Με το κατηγόρημα solution/2 ελέγχουμε αν η λίστα στο input/1 μπορεί να ταξινομηθεί με μια ακολουθία κινήσεων S με μήκος το πολύ μέχρι Max.

Συγκεκριμένα στο σώμα της solution/2 αρχικά ορίζουμε την L ως την λίστα που δίνεται στο input, ταξινομούμε την L και αποθηκεύουμε το αποτέλεσμα στην λίστα L1 (το κατηγόρημα sort/2 δίνεται από την SWI Prolog).

Στην συνέχεια με το κατηγόρημα belongs/3 παράγουμε (generate) μια ακολουθία κινήσεων S που έχει μήκος το πολύ Max, και που αποτελείται από αριθμούς που βρίσκονται στην L.

Τέλος μέσω του turnSequence/3 ελέγχουμε (test) αν ξεκινώντας από την L και με την διαδοχική εκτέλεση της ακολουθίας κινήσεων S, καταλήγουμε στην ταξινομημένη λίστα L1.

Τα κατηγορήματα belongs/3 και turnSequence/3 θα πρέπει να τα ορίσετε εσείς με τον τρόπο που περιγράφεται στην συνέχεια.

Α. [2 *μονάδες*] Συμπληρώστε τον παρακάτω κώδικα για να ορίσετε το κατηγόρημα belongs/3 έτσι ώστε το belongs(S,L,Max)να αληθεύει όταν η λίστα S έχει το πολύ Max στοιχεία, και όλα τα στοιχεία της S ανήκουν στην λίστα L. [Υπόδειξη: Χρησιμοποιείστε αναδρομή και το κατηγόρημα member/2 της SWI Prolog].

```
belongs([],_,Max) :- 0=<Max.
belongs([A|B],L,Max) :- .....
```

Απάντηση:

```
belongs([], ,Max) :- 0=<Max.
belongs([A|B],L,Max) :- length([A|B],N), N=<Max, Max1 is Max-1, member(A,L),belongs(B,L,Max1).
```

Επεξήγηση:

Πρέπει όλα τα στοιχεία της πρώτης λίστας να ανήκουν στην δεύτερη λίστα και η πρώτη λίστα να έχει το πολύ MAX στοιχεία (μπορεί και λιγότερα, όχι περισσότερα).

Η βασική περίπτωση λέει: τα στοιχεία της κενής λίστας (το κανένα στοιχείο δηλαδή) ανήκουν σίγουρα σε οποιαδήποτε λίστα (ανώνυμη μεταβλητή) και αρκεί απλά το πλήθος στοιχείων MAX να είναι μεγαλύτερο ή ίσο του μηδενός. Πράγματι η κενή λίστα με 0 στοιχεία έχει πλήθος στοιχείων το πολύ MAX, αρκεί το MAX να είναι 0 ή θετικό. Η αναδρομική περίπτωση λέει: πρέπει αρχικά να ελέγξω ότι το πλήθος των στοιχείων της λίστας είναι το πολύ MAX, μετά να ελέγξω αν το πρώτο στοιχείο A της λίστας ανήκει στην λίστα L. Αυτό κάνει η member(A,L). Στην συνέχεια, αναδρομικά καλώ την belongs για την υπόλοιπη λίστα στοιχείων B που θέλω επίσης να ανήκουν στην λίστα L. Μειώνω τον μετρητή Max κατά 1 (γι αυτό φτιάχνω την νέα μεταβλητή Max1 που ισούται με Max-1) για να δείξω ότι πλέον έχω μετρήσει ένα στοιχείο (το A), άρα απομένουν το πολύ Max-1 στοιχεία.

[2] Μάλιστα υπάρχουν στοίβες για τις οποίες υπάρχουν μόνο τέτοιες λύσεις. Για παράδειγμα η στοίβα [4,1,3,5,2] δεν μπορεί να ταξινομηθεί αν δεν επιτρέπεται η τοποθέτηση της σπάτουλας κάτω από την ίδια πίτα πάνω από μια φορά. Οι μόνες ακολουθίες κινήσεων που ταξινομούν την στοίβα, όπως η [3,1,5,2,1], τοποθετούν την σπάτουλα *δύο (ή περισσότερες) φορές* κάτω από την ίδια πίτα.

Β. [3 *μονάδες*] Συμπληρώστε τον παρακάτω κώδικα για να ορίσετε ένα κατηγόρημα turnOnce/3, τέτοιο ώστε το turnOnce(N,In,Out) να αληθεύει όταν η λίστα Out προκύπτει από την λίστα In με *μια* κίνηση της σπάτουλας όταν αυτή τοποθετείται κάτω από την πίτα N. Επιπλέον, για να αποφεύγονται περιττές κινήσεις, θα πρέπει το N να είναι διαφορετικό από το πρώτο στοιχεία της In. Για παράδειγμα το turnOnce(4,[2,4,1,3,5],[4,2,1,3,5])είναι αληθές.

turnOnce(N, In, Out) :- append(A, [N|B], In), length(A,K), 0<K, ...

[Υπόδειξη:
Η βασική ιδέα στην υλοποίηση της turnOnce(N, In, Out) είναι η εξής:

1. Διαχωρίζεται την λίστα In σε δύο υπο-λίστες, εκ των οποίων η πρώτη υπο-λίστα, με το όνομα A, περιέχει όλα τα στοιχεία της In πριν από το N, και η δεύτερη υπο-λίστα, [N|B], περιέχει όλα τα υπόλοιπα στοιχεία της In. Αυτός ο διαχωρισμός μπορεί να γίνει με το κατηγόρημα append/3 της SWI Prolog όπως φαίνεται στον παραπάνω κανόνα.
2. Αναποδογυρίζετε τις πίτες της A με το κατηγόρημα της SWI Prolog reverse/2.
3. Το αποτέλεσμα το ενώνετε με την λίστα B μέσω του κατηγορήματος append/3, τοποθετώντας παράλληλα την πίτα N στην κορυφή της στοίβας. Την νέα στοίβα την ονομάζεται Out.]

Επισημαίνεται ότι τα length(A,K), 0<K έχουν προστεθεί στο σώμα του παραπάνω κανόνα για να αποφευχθούν οι περιττές κινήσεις.[3] Συγκεκριμένα, το length(A,K) αποθηκεύει στο K το πλήθος των στοιχείων της λίστας A. Επομένως το 0<K εξασφαλίζει πως η λίστα A δεν είναι κενή, ή αλλιώς πως η σπάτουλα δεν έχει τοποθετηθεί κάτω από την πρώτη πίτα στην κορυφή της λίστας (κάτι τέτοιο δεν θα επέφερε καμία μεταβολή στην στοίβα – είναι περιττή κίνηση).

Απάντηση:

```
turnOnce(N, In, Out) :- append(A, [N|B], In), length(A,K), 0<K, reverse(A,ReversedA), append([N|ReversedA],B,Out).
```

Επεξήγηση:

Το να γυρίσω την στοίβα IN με πίτες και να προκύψει η στοίβα OUT βάζοντας την σπάτουλα στην πίτα N σημαίνει να κάνω τα εξής βήματα: χωρίζω την στοίβα IN στο σημείο που είναι η πίτα N όπου θα βάλω την σπάτουλα. Άρα A είναι η στοίβα με τις πίτες πάνω από την πίτα N και B είναι η στοίβα με τις πίτες κάτω από την πίτα N. Η στοίβα A με τις πίτες θα αναποδογυρίσει, αυτό κάνει το κατηγόρημα reverse. Η λίστα ReversedA είναι οι πίτες της A αναποδογυρισμένες. Η στοίβα B με τις κάτω πίτες δεν θα αναποδογυρίσει, θα μείνει ως έχει. Συνεπώς, πάνω-πάνω θα μπει η πίτα N, μετά η αναποδογυρισμένη στοίβα ReversedA, μετά η στοίβα B και έτσι προκύπτει ως αποτέλεσμα η στοίβα OUT. Αυτό επιτυγχάνει η τελευταία append.

Γ. [2 *μονάδες*] Συμπληρώστε τον παρακάτω κώδικα για να ορίσετε ένα κατηγόρημα turnSequence/3, τέτοιο ώστε το turnSequence(S,In,Out) να αληθεύει όταν ξεκινώντας από την στοίβα In και εκτελώντας διαδοχικά τις κινήσεις στην λίστα S, καταλήγουμε στην στοίβα Out. Για παράδειγμα το turnSequence([3,1],[2,4,1,3,5],[1,3,4,2,5])είναι αληθές. [Υπόδειξη: Χρησιμοποιήστε αναδρομή και το κατηγόρημα turnOnce/3 που ορίσατε προηγουμένως.]

turnSequence([],L,L).
turnSequence([N|L], In, Out) :- .....

Απάντηση:

```
turnSequence([],L,L).
turnSequence([N|L], In, Out) :- turnOnce(N,In,Out1), turnSequence(L,Out1,Out).
```

Επεξήγηση:

Πρέπει στην στοίβα IN με πίτες να εφαρμόσω διαδοχικά τα γυρίσματα στις πίτες που περιέχονται στην πρώτη λίστα. Έτσι προκύπτει η τελική στοίβα OUT με πίτες. Η βασική περίπτωση λέει: αν δεν έχω κανένα άλλο γύρισμα να εφαρμόσω στην στοίβα L, τότε το αποτέλεσμα είναι προφανώς η στοίβα L. Η αναδρομική περίπτωση λέει: πρέπει να εφαρμόσω το πρώτο αναποδογύρισμα στην πίτα N και μετά αναδρομικά/διαδοχικά τα αναποδογυρίσματα στις υπόλοιπες πίτες τις λίστας L. Συνεπώς, εφαρμόζω το πρώτο αναποδογύρισμα στην πίτα N καλώντας της turnOnce(N,In,Out1) και μετά το πρώτο αυτό αναποδογύρισμα προκύπτει ως αποτέλεσμα η στοίβα Out1. Στην στοίβα Out1 εφαρμόζω αναδρομικά τα υπόλοιπα αναποδογυρίσματα στις πίτες που αναφέρει η λίστα L και έτσι προκύπτει τελικά (αναδρομικά) η λίστα Out.

[3] Το `length/2` είναι κατηγόρημα της SWI Prolog.

Δ1. [1 *μονάδα*] Εκτελέστε το query solution(S,3) για να διαπιστώσετε αν υπάρχει λύση με το πολύ 3 κινήσεις για την στοίβα στα αριστερά του Σχήματος 1; Γράψτε την απάντηση που δίνει το πρόγραμμά σας.
Απάντηση:
Δεν λειτούργησε πλήρως ο κώδικας, συνεπώς δεν θα απαντηθεί

Δ2. [1 *μονάδα*] Ορίστε ως input την στοίβα [5, 3, 6, 2, 4, 1, 7]. Χρησιμοποιήστε το πρόγραμμά σας για να βρείτε τον ελάχιστο αριθμό κινήσεων που θα την ταξινομήσει. Ποια λύση δίνει το πρόγραμμά σας γι' αυτό τον αριθμό κινήσεων;
Απάντηση:
Δεν λειτούργησε πλήρως ο κώδικας, συνεπώς δεν θα απαντηθεί

Δ3. [1 *μονάδα*] Ορίστε ως input την στοίβα [5, 3, 6, 2, 9, 4, 1, 8, 7] και ελέγξτε αν αυτή η στοίβα μπορεί να ταξινομηθεί με το πολύ Ν κινήσεις. Ποιος είναι ο μέγιστος αριθμός κινήσεων Ν για τον οποίο το πρόγραμμά σας δίνει (θετική ή αρνητική) απάντηση σε λιγότερο από 2 λεπτά πραγματικού χρόνου; Βρήκατε κάποια ακολουθία κινήσεων που ταξινομεί την λίστα μέσα σ' αυτόν τον χρονικό περιορισμό;
Απάντηση:
Δεν λειτούργησε πλήρως ο κώδικας, συνεπώς δεν θα απαντηθεί

Ο κώδικας είναι το αρχείο thema5.pl