ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εαρινό 2021

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 2

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 30/5/2021, 23:59:59

ΜΕΘ υπό διαρκή πίεση, ξανά (0.25 βαθμοί)

Το πρόβλημα αυτό είναι γνωστό από την πρώτη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του σε Prolog. Επειδή τα συστήματα Prolog δεν τρέχουν native code, ο χρονικός περιορισμός για την άσκηση θα είναι σημαντικά αυξημένος. Το πρόγραμμά σας θα πρέπει να περιέχει ένα κατηγόρημα **`longest/2`** το οποίο θα έχει ως πρώτο όρισμα το όνομα του αρχείου εισόδου και θα επιστρέφει στο δεύτερο όρισμά του την απάντηση. Για το παράδειγμα της εκφώνησης της πρώτης σειράς, το κατηγόρημά σας θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω.[1]

```
?- longest('f.txt', Answer), writeln(Answer), fail.
5
false.
```

Για το διάβασμα της εισόδου, δείτε το υπόδειγμα που δίνεται στη δεύτερη άσκηση.

Βρόχοι σε τηλεπαιχνίδια, ξανά (0.25 βαθμοί)

Και αυτό το πρόβλημα είναι γνωστό από την πρώτη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του σε Python. Επειδή οι υλοποιήσεις της Python δεν τρέχουν native code, ο χρονικός περιορισμός για αυτή την άσκηση θα είναι και πάλι αυξημένος. Το πρόγραμμά σας θα πρέπει να έχει την ίδια συμπεριφορά με τα προγράμματα σε C/C++ που παραδώσατε για την πρώτη σειρά ασκήσεων. Για τα παραδείγματα της εκφώνησης της πρώτης σειράς, η έξοδος του προγράμματός σας πρέπει να είναι η εξής:

```
$ python3 loop_rooms.py maze1.txt
4

$ python3 loop_rooms.py maze2.txt
2
```

Ταξινόμηση με ουρά και στοίβα (0.25+0.25 = 0.5 βαθμοί)

Δίνονται *N* μη αρνητικοί το πολύ τριψήφιοι ακέραιοι αριθμοί (όπου $1 \le N \le 42$)[2], οι οποίοι τοποθετούνται σε μία ουρά. Δίνεται επίσης μία στοίβα, που αρχικά είναι κενή. Ο σκοπός αυτής της άσκησης είναι να ταξινομήσετε τους αριθμούς που βρίσκονται στην ουρά σε αύξουσα σειρά, χρησιμοποιώντας μία ακολουθία αποτελούμενη από τις εξής κινήσεις:

- **Q**: Αφαίρεσε το πρώτο στοιχείο της ουράς και τοποθέτησέ το στην κορυφή της στοίβας.
- **S**: Αφαίρεσε το στοιχείο που βρίσκεται στην κορυφή της στοίβας και τοποθέτησέ το στο τέλος της ουράς.

---

[1] Σε όλα τα παραδείγματα αυτής της σειράς ασκήσεων, ανάλογα με το σύστημα Prolog που θα χρησιμοποιήσετε, στις περιπτώσεις που υπάρχει κάποια λύση, η γραμμή με το **`false.`** μπορεί να λέει **`fail.`** ή **`no.`** Επίσης, προσέξτε ότι με τη χρήση **`writeln`** και **`fail`**, όπως φαίνεται στα παραδείγματα, μπορείτε να καταλάβετε και αν η εκτέλεση του κυρίως κατηγορήματος του προγράμματός σας είναι ντετερμινιστική ή όχι.

[2] Το άνω όριο ($N \le 42$) μην το πάρετε τοις μετρητοίς. Τα test cases θα επιτρέπουν την εξαντλητική αναζήτηση.

Οι κινήσεις της ακολουθίας εφαρμόζονται η μία μετά την άλλη κατά σειρά και μετά την εκτέλεση της τελευταίας θα πρέπει η στοίβα να είναι κενή και οι αριθμοί που βρίσκονται στην ουρά να είναι ταξινομημένοι σε αύξουσα σειρά, με τον μικρότερο αριθμό στην αρχή της ουράς. Αν υπάρχουν περισσότερες ακολουθίες κινήσεων που οδηγούν στο επιθυμητό αποτέλεσμα, μας ενδιαφέρει να βρούμε αυτή που έχει το μικρότερο μήκος (δηλαδή το μικρότερο πλήθος κινήσεων). Αν υπάρχουν περισσότερες ακολουθίες ίσου μήκους, μας ενδιαφέρει να βρούμε τη λεξικογραφικά μικρότερη.

Για παράδειγμα, έστω ότι N=4 και ότι οι αριθμοί 7, 17, 3, 42 βρίσκονται αρχικά στην ουρά, με αυτή τη σειρά. Η ακολουθία 10 κινήσεων “QQSQSSQQSS” πετυχαίνει το επιθυμητό αποτέλεσμα, όπως φαίνεται στον παρακάτω πίνακα (η αρχή της ουράς βρίσκεται αριστερά και το τέλος της δεξιά, ενώ η κορυφή της στοίβας βρίσκεται δεξιά):

Βήμα	Κίνηση	Κατάσταση	
0	αρχικά	ουρά: στοίβα:	7, 17, 3, 42 (κενή)
1	εκτελείται Q	ουρά: στοίβα:	17, 3, 42 7
2	εκτελείται Q	ουρά: στοίβα:	3, 42 7, 17
3	εκτελείται S	ουρά: στοίβα:	3, 42, 17 7
4	εκτελείται Q	ουρά: στοίβα:	42, 17 7, 3
5	εκτελείται S	ουρά: στοίβα:	42, 17, 3 7

Βήμα	Κίνηση	Κατάσταση	
6	εκτελείται S	ουρά: στοίβα:	42, 17, 3, 7 (κενή)
7	εκτελείται Q	ουρά: στοίβα:	17, 3, 7 42
8	εκτελείται Q	ουρά: στοίβα:	3, 7 42, 17
9	εκτελείται S	ουρά: στοίβα:	3, 7, 17 42
10	εκτελείται S	ουρά: στοίβα:	3, 7, 17, 42 (κενή)

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε Prolog και ένα σε Python) τα οποία θα απαντούν στο παραπάνω ερώτημα. Οι αριθμοί που τοποθετούνται αρχικά στην ουρά θα θεωρούνται γνωστοί και η απάντηση των προγραμμάτων σας μπορεί να βασίζεται σε κατάλληλη επεξεργασία αυτών των αριθμών. Επειδή οι υλοποιήσεις της Prolog και της Python δεν τρέχουν native code, ο χρονικός περιορισμός για τα πρόγραμματά σας θα είναι αυξημένος.

Η είσοδος των προγραμμάτων σας διαβάζεται από ένα αρχείο κειμένου που θα περιέχει δύο γραμμές. Η πρώτη γραμμή θα περιέχει το πλήθος *N* των αριθμών, ενώ η δεύτερη θα περιέχει τους *N* αριθμούς, χωρισμένους ανά δύο με ένα κενό διάστημα. Η έξοδος των προγραμμάτων σας πρέπει να είναι η ζητούμενη ακολουθία κινήσεων. Αν αυτή είναι κενή, η έξοδος θα πρέπει να είναι η λέξη “empty”. Παρακάτω δίνονται παραδείγματα, σε Prolog και Python.

Σε Prolog

```
?- longest('qs1.txt', Answer), writeln(Answer), fail.
QQSQSSQQSS
false.
?- longest('qs2.txt', Answer), writeln(Answer), fail.
QQQQSQSSSQSSQS
false.
?- longest('qs3.txt', Answer), writeln(Answer), fail.
QQSQQSQSSS
false.
?- longest('qs4.txt', Answer), writeln(Answer), fail.
QQQQSSSS
false.
?- longest('qs5.txt', Answer), writeln(Answer), fail.
empty
false.
```

Σε Python

```
$ python3 qssort.py qs1.txt
QQSQSSQQSS
$ python3 qssort.py qs2.txt
QQQQSQSSSQSSQS
$ python3 qssort.py qs3.txt
QQSQQSQSSS
$ python3 qssort.py qs4.txt
QQQQSSSS
$ python3 qssort.py qs5.txt
empty
```

```
$ cat qs1.txt
4
7 17 3 42
$ cat qs2.txt
6
17 7 3 42 1 8
$ cat qs3.txt
6
1 0 1 0 1 0
$ cat qs4.txt
5
5 4 3 2 1
$ cat qs5.txt
4
1 2 3 4
```

Το πρώτο παράδειγμα είναι αυτό που περιγράφεται παραπάνω. Προσέξτε ότι και η ακολουθία κινήσεων "QSQSQSQQSQSS" επιτυγχάνει το επιθυμητό αποτέλεσμα, είναι όμως μακρύτερη (12 κινήσεις αντί 10). Στο τρίτο παράδειγμα, η ακολουθία κινήσεων "QQSQQSSSQS" επίσης επιτυγχάνει το επιθυμητό αποτέλεσμα, είναι όμως ίσου μήκους (10 κινήσεων) και η "QQSQQSQSSS" είναι λεξικογραφικά μικρότερη. Στο πέμπτο παράδειγμα, οι αριθμοί είναι ήδη ταξινομημένοι.

Περαιτέρω οδηγίες για τις ασκήσεις

- Μπορείτε να δουλέψετε σε ομάδες το πολύ δύο ατόμων. Μπορείτε αν θέλετε να σχηματίσετε διαφορετική ομάδα σε σχέση με την προηγούμενη σειρά ασκήσεων – οι ομάδες στο σύστημα υποβολής είναι έτσι και αλλιώς καινούργιες για κάθε σειρά ασκήσεων.

- Δεν επιτρέπεται να μοιράζεστε τα προγράμματά σας με συμφοιτητές εκτός της ομάδας σας ή να τα βάλετε σε μέρος που άλλοι μπορούν να τα βρουν (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοσελίδες συζητήσεων, …). Σε περίπτωση που παρατηρηθούν «περίεργες» ομοιότητες σε προγράμματα, ο βαθμός των εμπλεκόμενων φοιτητών σε *όλες τις σειρές ασκήσεων* γίνεται αυτόματα μηδέν ανεξάρτητα από το ποια ομάδα... «εμπνεύστηκε» από την άλλη.

- Μπορείτε να χρησιμοποιήσετε «βοηθητικό» κώδικα (π.χ. κάποιο κώδικα που διαχειρίζεται κάποια δομή δεδομένων) που βρήκατε στο διαδίκτυο στα προγράμματά σας, με την προϋπόθεση ότι το πρόγραμμά σας περιέχει σε σχόλια την παραδοχή για την προέλευση αυτού του κώδικα και ένα σύνδεσμο σε αυτόν.

- Τα προγράμματα σε Python πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε Python 3.7.3. (Προσέξτε ότι η Python 2 είναι διαφορετική διάλεκτος της Python!)

- Τα προγράμματα σε Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog (8.0.2), GNU Prolog (1.3.0) ή YAP (6.2.2).

- Η υποβολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του moodle, όπως και στην προηγούμενη άσκηση, και για να μπορέσετε να τις υποβάλλετε, τα μέλη της ομάδας σας (και οι δύο) θα πρέπει να έχουν ήδη λογαριασμό στο moodle. Θα υπάρξει σχετική ανακοίνωση μόλις το σύστημα υποβολής καταστεί ενεργό. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο εκτός από τη ζητούμενη διότι δε θα γίνουν δεκτά από το σύστημα υποβολής.