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

Εαρινό 2020

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 2

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 17/5/2020, 23:59:59

Δυνάμεις του δύο, ξανά (0.25 βαθμοί)

Το πρόβλημα με τις δυνάμεις του δύο 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 **`powers2/2`** το οποίο θα έχει ως πρώτο όρισμα το όνομα του αρχείου εισόδου και θα επιστρέφει στο δεύτερο όρισμά του τη λίστα με τις απαντήσεις. Για τα παραδείγματα της εκφώνησης της πρώτης σειράς, το κατηγόρημά σας θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω.[1]

```
?- powers2('f.txt', Answers), writeln(Answers), fail.
[[1,2,1],[3,1,1],[],[2,2,1,0,0,1]]
false.
```

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

Κορωνογράφοι, ξανά (0.25 βαθμοί)

Το πρόβλημα με τον εντοπισμό κορωνογράφων είναι κι αυτό γνωστό από την πρώτη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο αυτής της άσκησης είναι και πάλι να γραφεί η λύση του σε Prolog. Το πρόγραμμά σας θα πρέπει να περιέχει ένα κατηγόρημα **`coronograph/2`** το οποίο θα έχει ως πρώτο όρισμα το όνομα του αρχείου εισόδου και θα επιστρέφει στο δεύτερο όρισμά του τη λίστα με τις απαντήσεις. Για τα παραδείγματα της εκφώνησης της πρώτης σειράς, το κατηγόρημά σας θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω.

```
?- coronograph('graphs.txt', Answers), writeln(Answers), fail.
[[3,[2,3,4]],'NO CORONA']
false.
```

Επιστροφή στο σπίτι (0.25+0.25 = 0.5 βαθμοί)

Το lockdown βρήκε το φίλο σας το Σωτήρη στο γραφείο του στο Πανεπιστήμιο και πρέπει να τον βοηθήσετε να επιστρέψει με ασφάλεια στο σπίτι του. Για να είναι ασφαλής ο Σωτήρης, θα πρέπει να αποφεύγει τις τοποθεσίες που είναι μολυσμένες από τον κορωνοϊό. Αυτό δεν είναι εύκολο γιατί η μόλυνση εξαπλώνεται μεταξύ γειτονικών τοποθεσιών και γιατί οι αεροπορικές μετακινήσεις δεν έχουν σταματήσει!

Δίνεται ένας διδιάστατος χάρτης της υφηλίου, αποτελούμενος από $N \times M$ τετραγωνάκια (όπου $2 \le N, M \le 1000$). Κάθε τετραγωνάκι του χάρτη περιέχει ένα από τα εξής σύμβολα:

"**`S`**": Η αρχική θέση του Σωτήρη, στο γραφείο του.

---

[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.

- "**T**": Η επιθυμητή τελική θέση του Σωτήρη, στο σπίτι του.
- "**W**": Το τετράγωνο από το οποίο ξεκινά η μόλυνση, κάπου στα βάθη της Ασίας.
- "**A**": Σε αυτό το τετράγωνο υπάρχει αεροδρόμιο.
- "**.**": (τελεία) Το τετράγωνο είναι κενό.
- "**X**": Σε αυτό το τετράγωνο δεν μπορεί να πάει ούτε ο Σωτήρης ούτε η μόλυνση (εμπόδιο, π.χ. θάλασσα).

Ο κορωνοϊός εξαπλώνεται διαρκώς στα γειτονικά τετράγωνα (αριστερά, δεξιά, πάνω και κάτω), αν αυτά δεν είναι εμπόδια, με αποτέλεσμα όλο και μεγαλύτερες περιοχές του χάρτη να μολύνονται. Η μόλυνση κινείται με ρυθμό ένα τετράγωνο ανά δύο μονάδες χρόνου. Αν φτάσει σε ένα αεροδρόμιο, τότε μετά από 5 μονάδες χρόνου θα πρέπει να θεωρήσουμε ότι έχει φτάσει σε όλα τα αεροδρόμια του κόσμου! Ευτυχώς για εκείνον, ο Σωτήρης είναι πιο γρήγορος: μπορεί να κινείται στα γειτονικά του τετράγωνα, με ρυθμό ένα τετράγωνο ανά μονάδα χρόνου, δεν έχει όμως καμία όρεξη μπαίνει σε αεροπλάνα.

Αυτό που πρέπει να βρείτε είναι ποια διαδρομή πρέπει να ακολουθήσει για να φτάσει στο σπίτι του με ασφάλεια και στον ελάχιστο δυνατό χρόνο (αν φυσικά μπορεί να συμβεί αυτό). Η απάντηση σε αυτή την ερώτηση είναι μία συμβολοσειρά η οποία περιγράφει την ακολουθία κινήσεων που θα κάνει ο Σωτήρης. Οι δυνατές κινήσεις παριστάνονται με τα σύμβολα:

- "**R**": Μετακίνηση ένα τετράγωνο δεξιά στο χάρτη.
- "**L**": Μετακίνηση ένα τετράγωνο αριστερά στο χάρτη.
- "**U**": Μετακίνηση ένα τετράγωνο προς τα πάνω στο χάρτη.
- "**D**": Μετακίνηση ένα τετράγωνο προς τα κάτω στο χάρτη.

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε ML και ένα σε Python) τα οποία θα απαντούν στα παραπάνω ερωτήματα. Επειδή οι υλοποιήσεις της Python δεν τρέχουν native code, ο χρονικός περιορισμός για το πρόγραμμά σας σε Python θα είναι αυξημένος σε σχέση με αυτόν σε ML.

Η είσοδος του προγράμματός σας διαβάζεται από ένα αρχείο αποτελούμενο από *N* γραμμές, κάθε μία από τις οποίες περιέχει *M* σύμβολα. Το αρχείο αυτό αναπαριστά το χάρτη.

Παρακάτω δίνονται κάποια παραδείγματα σε ML και Python. Έστω ότι τα αρχεία με τα δεδομένα εισόδου είναι τα εξής (όπως είπαμε, η εντολή **cat** είναι εντολή του Unix και από μόνη της δεν συνηθίζει να χρωματίζει χάρτες της υφηλίου):

```
$ cat s1.txt
S.........XAW
A.........XXX
............
............
......A....T
```

```
$ cat s2.txt
.........
.XXX...XA
.XAWX....
S..X...T.
....XXX..
.........
```

```
$ cat s3.txt
.....
.X.X.
SXWXT
```

Η έξοδος του προγράμματός σας πρέπει να είναι η ακόλουθη. Στην πρώτη γραμμή πρέπει να τυπώνεται ο ελάχιστος χρόνος στον οποίο ο Σωτήρης μπορεί να βρεθεί με ασφάλεια στο σπίτι του ή, αν αυτό δεν μπορεί να συμβεί, η λέξη "**IMPOSSIBLE**". Σε περίπτωση που είναι εφικτό να φτάσει ο Σωτήρης στο σπίτι του, θα πρέπει να εκτυπώνεται και μία δεύτερη γραμμή που να περιέχει μία συμβολοσειρά: την ακολουθία των κινήσεων που πρέπει να κάνει ο Σωτήρης. (Φυσικά, το μήκος αυτής της συμβολοσειράς θα είναι ίσο με τον ακέραιο αριθμό που εκτυπώνεται στην πρώτη γραμμή.) Αν υπάρχουν πολλές διαφορετικές λύσεις, τότε επιλέγουμε τη λεξικογραφικά μικρότερη.

Σε MLton ή OCaml

```
$ ./stayhome s1.txt
15
DDRRRRRRRDRRRDR

$ ./stayhome s2.txt
12
UUURRRRDRDDR

$ ./stayhome s3.txt
IMPOSSIBLE
```

Σε SML/NJ

```
- stayhome "s1.txt";
15
DDRRRRRRRDRRRDR
val it = () : unit
- stayhome "s2.txt";
12
UUURRRRDRDDR
val it = () : unit
- stayhome "s3.txt";
IMPOSSIBLE
val it = () : unit
```

Σε Python

```
$ python3 stayhome.py s1.txt
15
DDRRRRRRRDRRRDR

$ python3 stayhome.py s2.txt
12
UUURRRRDRDDR

$ python3 stayhome.py s3.txt
IMPOSSIBLE
```

Στο πρώτο παράδειγμα ο κορωνοϊός φτάνει στο αεροδρόμιο κοντά στο σπίτι αλλά ο Σωτήρης είναι γρηγορότερος. Μπορεί να φτάσει στο σπίτι του σε 15 κινήσεις, ενώ ο ιός θα φτάσει εκεί σε 17 — παραλίγο! Για να πετύχει τη λεξικογραφικά ελάχιστη δυνατή ακολουθία κινήσεων θα έπρεπε να πάει πρώτα κάτω και μετά δεξιά, αυτό όμως δε γίνεται γιατί πρέπει να αποφεύγει τα μολυσμένα τετράγωνα. Μελετήστε τον ελιγμό του, μια χαρά τα κατάφερε...

Στο δεύτερο παράδειγμα, αν δεν υπήρχε ο ιός, ο Σωτήρης θα μπορούσε να πάει στο σπίτι του σε 12 κινήσεις, είτε από την πάνω διαδρομή είτε από την κάτω, και λεξικογραφικά μικρότερη είναι η κάτω. Όμως, με την εξάπλωση του ιού, από τις δύο διαδρομές μόνο η πάνω είναι εφικτή, λόγω του αεροδρομίου που βρίσκεται πάνω δεξιά.

Στο τρίτο παράδειγμα δεν είναι δυνατό ο Σωτήρης να φτάσει στο σπίτι του γιατί θα περάσει από μολυσμένο τετράγωνο τη χρονική στιγμή $t = 4$.

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 ML πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε SML/NJ ≥ v110.79

ή σε MLton ≥ 20130715 ή σε Objective Caml version ≥ 4.05.0. Το σύστημα ηλεκτρονικής υποβολής σας επιτρέπει να επιλέξετε μεταξύ αυτών των διαλέκτων της ML.

- 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.