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

Εαρινό 2017

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 23/7/2017, 23:59:59
ή (αν δεν θέλετε βαθμό τον Ιούλιο): 20/8/2017, 23:59:59

Εκδρομή για σκι (0.25 βαθμοί)

Το πρόβλημα με την Κατερίνα και το σκι είναι γνωστό από την πρώτη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του προβλήματος 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 **`skitrip/2`** το οποίο θα έχει ως πρώτο όρισμα το όνομα του αρχείου εισόδου και θα επιστρέφει στο δεύτερό του όρισμα τη λύση. Για το παραδείγματα της εκφώνησης της άσκησης, το κατηγόρημά σας θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω.[1]

```
?- skitrip('h1.txt', Answer).
Answer = 10 ;
false.

?- skitrip('h2.txt', Answer).
Answer = 0 ;
false.
```

Για το διάβασμα της εισόδου, δείτε το υπόδειγμα που δίνεται στην τρίτη άσκηση.

Ο Λάκης αντεπιτίθεται (0.25 βαθμοί)

Το πρόβλημα με τον Λάκη τον εξωγήινο (στη δεύτερη μορφή του) είναι γνωστό από τη δεύτερη σειρά ασκήσεων. Το ζητούμενο είναι να γραφεί η λύση του προβλήματος σε Prolog. Το κατηγόρημα **`moredeli/3`**, που το πρόγραμμά σας θα πρέπει να περιέχει, δέχεται τρία ορίσματα: το πρώτο είναι το όνομα του αρχείου που περιέχει το χάρτη, ενώ στα άλλα δύο επιστρέφονται αντίστοιχα το κόστος της λύσης και η λύση που βρήκατε. Η λύση θα πρέπει να είναι μία λίστα από άτομα, όπως φαίνεται στα παρακάτω:

```
?- moredeli('map1.txt', Cost, Solution).
Cost = 16,
Solution = [u, r, r, r, d, d, d, l, l, u].

?- moredeli('map2.txt', Cost, Solution).
Cost = 26,
Solution = [d, l, l, l, l, u, u, u, u, r, r, d, r, d].
```

Όπως και στην προηγούμενη άσκηση, το όριο του χρόνου εκτέλεσης θα είναι σημαντικά αυξημένο.

---

[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`** ή μπορεί να μην τυπώνεται (που μάλλον είναι το καλύτερο διότι δείχνει ότι η εκτέλεση του κατηγορήματός σας είναι ντετερμινιστική).

Πωλείται σκάλα σε τιμή ευκαιρίας (0.25+0.25 = 0.5 βαθμοί)

Έστω μια σκάλα που οι φήμες λένε ότι οδηγεί στον παράδεισο, αποτελούμενη από N σκαλοπάτια. Κάποια από αυτά είναι χαλασμένα και δυστυχώς δεν μπορούν να επισκευαστούν — ακόμα και στον παράδεισο ισχύουν τα capital controls. Ευτυχώς όλοι έχουν πια μάθει να πορεύονται με αυτό. Ξέρουμε ότι τα χαλασμένα σκαλοπάτια είναι ακριβώς B (όπου $0 \le B \le N$) και ότι το i-οστό από αυτά βρίσκεται στη θέση X_i (όπου $1 \le i \le B$ και $1 \le X_i \le N$).

Η γυναίκα της ιστορίας μας ξεκινάει από την αρχή της σκάλας, δηλαδή από το σκαλοπάτι με αριθμό 1, και ο σκοπός της είναι να βρεθεί στο σκαλοπάτι με αριθμό N. Σε κάθε βήμα της επιλέγει πόσα σκαλοπάτια θα ανέβει από τις τιμές ενός δοθέντος συνόλου $\Sigma = \{ S_1, \ldots S_K \}$, όπου $1 \le K \le N$ και $1 \le S_i < N$. Προφανώς δεν μπορεί να πατήσει σε χαλασμένο σκαλοπάτι (και αν είτε το σκαλοπάτι με αριθμό 1 είτε εκείνο με αριθμό N είναι χαλασμένα, τότε προφανώς η φίλη μας ατύχησε). Εσείς πρέπει να βρείτε με πόσους διαφορετικούς τρόπους (δηλαδή με πόσες διαφορετικές ακολουθίες βημάτων) μπορεί να πετύχει το σκοπό της. Επειδή ο αριθμός αυτός μπορεί να είναι υπερβολικά μεγάλος (και ο υπολογιστής σας μάλλον δεν είναι φτιαγμένος στον παράδεισο οπότε ενδέχεται να υπερχειλίσει), υπολογίστε το υπόλοιπο της ακέραιας διαίρεσης της σωστής απάντησης δια του αριθμού 1.000.000.009.

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε Prolog και ένα άλλο σε όποια γλώσσα επιθυμείτε από τις C/C++, ML ή Java) τα οποία να συμπεριφέρονται όπως στα επόμενα παραδείγματα. Στην Prolog, το κατηγόρημα **`hopping/3`** που πρέπει να περιέχει το πρόγραμμά σας θα δέχεται δύο ορίσματα: το πρώτο θα είναι το όνομα του αρχείου εισόδου, ενώ στο δεύτερο θα επιστρέφεται η απάντηση.

Σε C/C++, MLton, ή σε OCaml

```
$ ./hopping h1.txt
2

$ ./hopping h2.txt
0

$ ./hopping h3.txt
238310156
```

Σε SML/NJ

```
- hopping "h1.txt";
val it = 2 : int

- hopping "h2.txt";
val it = 0 : int

- hopping "h3.txt";
val it = 238310156 : int
```

Σε Prolog

```
?- hopping('h1.txt', Answer).
Answer = 2.

?- hopping('h2.txt', Answer).
Answer = 0.

?- hopping('h3.txt', Answer).
Answer = 238310156.
```

Σε Java

```
- java Hopping h1.txt
2

- java Hopping h2.txt
0

- java Hopping h3.txt
238310156
```

Τα στοιχεία εισόδου θα διαβάζονται από ένα αρχείο όπως φαίνεται στα παραδείγματα. Η πρώτη γραμμή του περιέχει τρεις ακέραιους αριθμούς N, K, B, χωρισμένους ανά δύο με ένα κενό διάστημα: το πλήθος N ($1 \le N \le 1.000.000$) των σκαλοπατιών, το πλήθος K των δυνατών βημάτων, και το πλήθος B των χαλασμένων σκαλοπατιών. Η δεύτερη γραμμή περιέχει ακριβώς K ακέραιους αριθμούς, χωρισμένους με ένα κενό διάστημα: τις τιμές των S_i (όπου $1 \le i \le K$). Η τρίτη γραμμή περιέχει ακριβώς Β ακέραιους αριθμούς, χωρισμένους με ένα κενό διάστημα: τις τιμές των X_i (όπου $1 \le i \le B$). Αν $B = 0$, η τρίτη γραμμή θα είναι κενή. Τα αρχεία με τα δεδομένα εισόδου για τα προηγούμενα παραδείγματα είναι τα εξής (η εντολή **`cat`** είναι εντολή του Unix):

```
$ cat h1.txt
10 3 4
1 2 4
2 4 6 7
```

```
$ cat h2.txt
10 2 4
4 1
6 8 2 9
```

```
$ cat h3.txt
100 2 0
1 2
```

Για το 1ο παράδειγμα, όπου τα βήματα είναι μήκους 1, 2, ή 4 σκαλοπατιών, υπάρχουν δύο τρόποι να ανέβει η φίλη μας τη σκάλα: $1 \to 3 \to 5 \to 9 \to 10$ και $1 \to 5 \to 9 \to 10$. Και οι δύο

αποφεύγουν τα χαλασμένα σκαλοπάτια 2, 4, 6 και 7. Για το 2ο παράδειγμα, δεν υπάρχει τρόπος η φίλη μας να πετύχει το σκοπό της — το πρώτο βήμα υποχρεωτικά πηγαίνει στο σκαλοπάτι 5 και μετά δεν υπάρχει επόμενο βήμα που να μην οδηγεί σε χαλασμένο σκαλοπάτι. Τέλος, στο 3ο παράδειγμα, τα δυνατά βήματα είναι μήκους 1 ή 2, η σκάλα αποτελείται από 100 σκαλοπάτια και κανένα δεν είναι χαλασμένο (προσέξτε ότι η τρίτη γραμμή στο αρχείο εισόδου υπάρχει αλλά είναι κενή). Εδώ, το πλήθος των τρόπων είναι ίσο με τον 100-στό αριθμό Fibonacci (πώς είπατε;) ο οποίος είναι ένας πολύ μεγάλος αριθμός:

354.224.848.179.261.915.075 = 354.224.844.991 × 1.000.000.009 + 238.310.156

Στην ιστοσελίδα του μαθήματος μπορείτε να βρείτε ένα αρχείο με δύο κατηγορήματα Prolog, το πρώτο από τα οποία διαβάζει αρχεία με δεδομένα εισόδου αυτής της άσκησης και σας τα επιστρέφει. Μπορείτε είτε να τα χρησιμοποιήσετε ως έχουν ή να τα τροποποιήσετε κατάλληλα για τις ανάγκες της λύσης σας.

Περαιτέρω οδηγίες για την άσκηση

- Μπορείτε να δουλέψετε σε ομάδες το πολύ δύο ατόμων. Μπορείτε αν θέλετε να σχηματίσετε διαφορετική ομάδα σε σχέση με τις προηγούμενες σειρές ασκήσεων – οι ομάδες στο σύστημα υποβολής είναι έτσι και αλλιώς καινούργιες για κάθε σειρά.
- 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 στις ασκήσεις 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 Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog (6.6.6), GNU Prolog (1.3.0) ή YAP (6.2.2).
- Τα προγράμματα στην άλλη γλώσσα μπορεί να είναι σε C/C++, ML (SML/NJ v110.76 ή MLton 20100608 ή Objective Caml version 4.01.0) ή σε Java. Το σύστημα ηλεκτρονικής υποβολής επιτρέπει να επιλέξετε μεταξύ αυτών των γλωσσών και των υλοποιήσεων της ML.
- Ο κώδικας των προγραμμάτων σε C/C++ και ML πρέπει να είναι σε ένα αρχείο. Ο κώδικας των προγραμμάτων σε Java μπορεί να βρίσκεται σε περισσότερα του ενός αρχείου αλλά θα πρέπει να μπορεί να μεταγλωττιστεί χωρίς προβλήματα με τον Java compiler με την εντολή **`javac Hopping.java`**
- Η υποβολή των προγραμμάτων θα γίνει ηλεκτρονικά όπως και στην προηγούμενη άσκηση. 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.