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

Εαρινό 2013

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 17/7/2013, 23:59:59

Το μεγάλο ψάρι τρώει το μικρό... (0.25 βαθμοί)

Το πρόβλημα με το παράλληλο σύμπαν και την οικονομική ένωση των χωρών είναι γνωστό από την πρώτη σειρά ασκήσεων. 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 αυξάνεται σε ένα λεπτό αντί για 10 secs. Το πρόγραμμά σας θα πρέπει να περιέχει ένα κατηγόρημα **`countries/3`** το οποίο, για τα παραδείγματα της εκφώνησης της πρώτης άσκησης, θα πρέπει να συμπεριφέρεται όπως φαίνεται παρακάτω[1].

```
?- countries('countries1.txt', Remaining, Expelled).
Remaining = 5
Expelled = 1 ;
false
?- countries('countries2.txt', Remaining, Expelled).
Remaining = 1
Expelled = 0 ;
false
?- countries('countries3.txt', Remaining, Expelled).
Remaining = 6
Expelled = 0 ;
false.
```

Ο Τάσος και τα πρόβατα... (0.25 βαθμοί)

Το πρόβλημα της μετακίνησης προβάτων στις στάνες του χωριού από τον Τάσο στις πλάτες του είναι γνωστό από τη δεύτερη σειρά ασκήσεων του μαθήματος. Το ζητούμενο είναι να γραφεί η λύση του προβλήματος σε Prolog. Κάποια παραδείγματα αντίστοιχα αυτών της δεύτερης σειράς από τη χρήση του κυρίου κατηγορήματος **`provata/2`** που το πρόγραμμά σας πρέπει να περιέχει φαίνονται παρακάτω.

```
?- provata('provata1.txt', MinCost).
MinCost = 8 ;
false.
?- provata('provata2.txt', MinCost).
MinCost = 13 ;
false.
?- provata('provata3.txt', MinCost).
MinCost = 20 ;
false.
```

[1] Σε όλα τα παραδείγματα αυτής της σειράς ασκήσεων, ανάλογα με το σύστημα Prolog που θα χρησιμοποιήσετε, η γραμμή με το **`false.`** μπορεί να λέει **`fail.`** ή **`no`** ή μπορεί να μην τυπώνεται.

Δυο πόρτες έχει η ζωή... (0.25+0.25 = 0.5 βαθμοί)

Σε αυτή την άσκηση όμως έχει πολύ περισσότερες. Ευτυχώς όμως, έχει και πολλά κλειδιά. Υπάρχουν πολλοί τύποι κλειδιών, αριθμημένοι με φυσικούς αριθμούς μικρότερους του 10^6. Κάθε πόρτα έχει πάνω της μία ή περισσότερες κλειδαριές που κάθε μία χρειάζεται έναν συγκεκριμένο τύπο κλειδιού για να ξεκλειδωθεί. Για να ανοίξει κανείς μία πόρτα, πρέπει να διαθέτει τα κατάλληλα κλειδιά για όλες τις κλειδαριές της. Επιπλέον, τα κλειδιά είναι μίας χρήσης: μόλις χρησιμοποιηθεί ένα κλειδί καταστρέφεται και δεν μπορεί να ξαναχρησιμοποιηθεί. Γιατί να θέλει κανείς να ανοίξει μία πόρτα; Μα φυσικά, γιατί πίσω της κρύβει κάτι που θέλουμε. Ανοίγοντας μία πόρτα, κερδίζουμε ένα χρηματικό ποσό στο αγαπημένο μας νόμισμα (το ευρώ είναι φυσικά ξεπερασμένο — άσε που σπανίζει τελευταία — οπότε ας υποθέσουμε ότι είναι bitcoins, *χιχι*, ομόλογα του ελληνικού δημοσίου, *μπουχαχα*, δικαιώματα στην ανακεφαλαιοποίηση της Εθνικής Τράπεζας, *αχαχαχαχα*, φασόλια ή οτιδήποτε άλλο θέλετε). Επιπλέον, ανοίγοντας μία πόρτα μπορεί να αποκτήσουμε κάποια κλειδιά που στη συνέχεια μπορούμε να χρησιμοποιήσουμε για να ανοίξουμε άλλες πόρτες.

Δίνεται μία λίστα αποτελούμενη από **K** κλειδιά που μας είναι διαθέσιμα στην αρχή του παιχνιδιού. Δίνεται επίσης μία λίστα από **G** πόρτες. Σε κάθε κίνηση, έχουμε δικαίωμα να ανοίξουμε μία πόρτα, αλλά φυσικά για να συμβεί αυτό πρέπει να έχουμε στη διάθεσή μας όλα τα κλειδιά που χρειάζονται. Μετά από κάθε κίνηση, η πόρτα που ανοίξαμε και τα κλειδιά που χρησιμοποιήσαμε εξαφανίζονται, πιστωνόμαστε το αντίστοιχο χρηματικό ποσό και τα κλειδιά που τυχον βρήκαμε προστίθενται στη λίστα των διαθέσιμων. Το παιχνίδι σταματάει όταν δεν μπορούμε να κάνουμε άλλη κίνηση, είτε γιατί δεν έχει μείνει καμία πορτα κλειστή, είτε γιατί δεν έχουμε διαθέσιμα κλειδιά για να ανοίξουμε καμία από τις κλειστές πόρτες.

Αυτό που ζητάει η άσκηση είναι να γραφούν δύο προγράμματα (ένα σε Prolog και το άλλο σε όποια γλώσσα επιθυμείτε από τις C, ML ή Java) τα οποία να παίρνουν ως είσοδο τις αρχικές λίστες των κλειδιών και των πορτών και να επιστρέφουν ως έξοδο έναν αριθμό: το μέγιστο χρηματικό ποσό που μπορούμε να έχουμε στα χέρια μας στο τέλος του παιχνιδιού.

Τα στοιχεία εισόδου θα διαβάζονται από ένα αρχείο με μορφή σαν και αυτή που φαίνεται στα παραδείγματα παρακάτω. Η πρώτη γραμμή του αρχείου έχει τον αριθμό **K** των κλειδιών που αρχικά έχουμε και τον αριθμό **G** των πορτών. Η επόμενη γραμμή περιέχει τα αρχικά κλειδιά και οι επόμενες **G** γραμμές αρχίζουν με τρεις αριθμούς $\mathbf{k}_i$, $\mathbf{r}_i$ και $\mathbf{s}_i$ — το πλήθος των κλειδιών που απαιτούνται για το άνοιγμα της πόρτας *i*, το πλήθος των κλειδιών που αποκτούμε ανοίγοντάς την, και το χρηματικό ποσό που κερδίζουμε, αντίστοιχα — και συνεχίζουν με $\mathbf{k}_i + \mathbf{r}_i$ κλειδιά.

Περιορισμοί: $1 \le \mathbf{K} \le 42$, $1 \le \mathbf{G} \le 42$, $1 \le \mathbf{k}_i \le 10$, $0 \le \mathbf{r}_i \le 10$, $0 < \mathbf{s}_i$, $\Sigma\mathbf{s}_i \le 10^9$. Όριο μνήμης: 4GB, χρόνου εκτέλεσης: 10 λεπτά (για Prolog), 1 λεπτό(για τις άλλες γλώσσες).

Παρακάτω δείχνουμε δύο παραδείγματα αρχείων εισόδου (η εντολή `cat` είναι εντολή του Unix):

```
> cat gates1.txt
1 4
1
1 0 200 1
1 2 150 1 1 3
2 0 120 2 1
1 2 140 3 1 2
```

```
> cat gates2.txt
3 5
1 3 1
2 1 100 1 3 2
3 2 900 1 1 2 5 3
2 2 200 5 1 1 3
1 2 130 2 3 5
2 2 120 3 2 1 1
```

Στην ιστοσελίδα του μαθήματος μπορείτε να βρείτε ένα πρόγραμμα Prolog που διαβάζει αρχεία αυτής της μορφής και σας δείχνει πώς μπορείτε να διαβάσετε ένα αρχείο με αριθμούς σε Prolog.

Το πρόγραμμά σας σε Prolog πρέπει να περιέχει ένα κατηγόρημα **gates/2** που να συμπεριφέρεται ως εξής:

```
?- gates('gates1.txt', MaxMoney).
MaxMoney = 610 ;
false.
?- gates('gates2.txt', MaxMoney).
MaxMoney = 430 ;
false.
```

Το πρόγραμμά σας σε SML/NJ πρέπει να έχει τη συμπεριφορά που φαίνεται παρακάτω:

```
- gates "gates1.txt";
val it = 610 : int
```

Το πρόγραμμά σας σε Java, C, MLton ή OCaml πρέπει να δουλεύει όπως φαίνεται παρακάτω:

```
$ java Gates gates1.txt
610
$ ./a.out gates1.txt
610
```

Περαιτέρω οδηγίες για την άσκηση

- 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 στις ασκήσεις γίνεται αυτόματα μηδέν ανεξάρτητα από το ποια ομάδα... «εμπνεύστηκε» από την άλλη.
- Τα προγράμματα σε Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog, GNU Prolog ή YAP.
- Τα προγράμματα στην άλλη γλώσσα μπορεί να είναι σε C, ML (SML/NJ v110.74 ή MLton 20100608 ή Objective Caml version 3.11.2) 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 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 Gates.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.