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

Εαρινό 2016

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 2

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 4/6/2016, 23:59:59

Δεξαμενές (0.25+0.25 = 0.5 βαθμοί)

Ο δήμαρχος ενός ακριτικού νησιού έχει εγκαταστήσει δεξαμενές σε διάφορα σημεία του νησιού. Οι δεξαμενές έχουν διαφορετικά σχήματα (και κατά συνέπεια όγκο) και βρίσκονται σε διαφορετικά ύψη από τη θάλασσα διότι το συγκεκριμένο νησί δεν είναι επίπεδο. Όλες όμως οι δεξαμενές είναι συνδεδεμένες μεταξύ τους με σωλήνες που τις τροφοδοτούν. Με δεδομένο ότι ξέρετε τα ύψη στα οποία είναι τοποθετημένες όλες οι δεξαμενές και τις διαστάσεις τους, αυτό που η άσκηση σας ζητάει να κάνετε είναι να γράψετε δύο προγράμματα (ένα σε ML και ένα σε Java) τα οποία να υπολογίζουν το μέγιστο ύψος από την επιφάνεια της θάλασσας στο οποίο θα φτάσει ένας συγκεκριμένος συνολικός όγκος νερού. Για απλοποίηση, θεωρήστε ότι οι σωλήνες που ενώνουν τις δεξαμενές συγκρατούν μέσα τους αμελητέα ποσότητα νερού.

Η είσοδος διαβάζεται από αρχεία. Η πρώτη γραμμή του κάθε αρχείου περιέχει τον αριθμό των δεξαμενών N ($1 \le N \le 420.000$) του νησιού. Οι επόμενες N γραμμές περιέχουν τέσσερις ακεραίους: B, H, W και L οι οποίοι αντιστοιχούν στο ύψος του επιπέδου της βάσης της κάθε δεξαμενής ($0 \le B \le 1.000.000$) και στις τρεις της διαστάσεις: ύψος, πλάτος και μήκος (για τα οποία ισχύει $1 \le H * W * L \le 42.000$). Η τελευταία γραμμή του αρχείου περιέχει τον όγκο του νερού με το οποίο θα γεμίσουν. Αν ο όγκος αυτός είναι μεγαλύτερος από τον συνολικό όγκο των δεξαμενών τότε συμβαίνει υπερχείλιση. Σε αυτήν την περίπτωση, το πρόγραμμά σας σε ML πρέπει να επιστρέφει **`~1.0`** και το πρόγραμμά σας σε Java πρέπει να τυπώνει **`Overflow`**. Αλλιώς, το πρόγραμμά σας θα πρέπει να επιστρέφει (στην ML) το ύψος μέχρι το οποίο θα γεμίσουν οι δεξαμενές, ή να τυπώνει (στη Java) το ύψος αυτό ως πραγματικό αριθμό στρογγυλεμένο (rounded) σε δύο μόνο δεκαδικά ψηφία. Τα παραδείγματα παρακάτω κάνουν όλα τα παραπάνω πιο σαφή.

Περιορισμοί: όριο χρόνου εκτέλεσης: 10 seconds, όριο μνήμης: 1 GB.

Παρακάτω δείχνουμε κάποιες πιθανές κλήσεις των προγραμμάτων σε ML και σε Java.

Σε SML/NJ

```
- deksamenes "data1.txt";
val it = 1.0 : real

- deksamenes "data2.txt";
val it = ~1.0 : real

- deksamenes "data3.txt";
val it = 17.0 : real
```

Σε Java

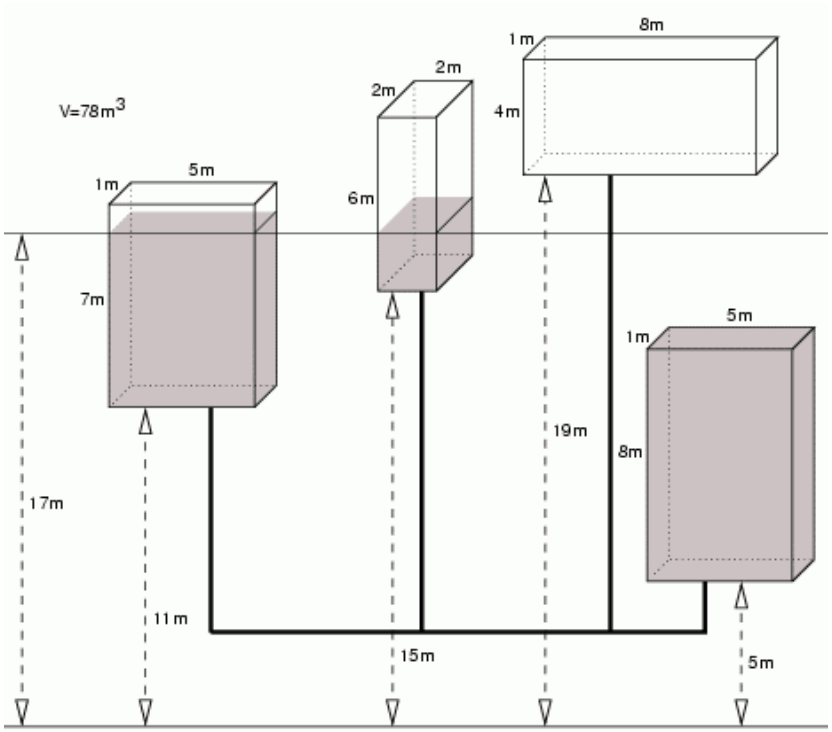
```
> java Deksamenes data1.txt
1.00

> java Deksamenes data2.txt
Overflow

> java Deksamenes data3.txt
17.00
```

όπου τα αρχεία εισόδου είναι αυτά που φαίνονται στην επόμενη σελίδα. Η εικόνα αντιστοιχεί στις δεξαμενές των **`data2`** και **`data3`** και το ύψος (17m) που δείχνει η εικόνα είναι αυτό για όγκο $78m^3$.

```
> cat data1.txt
2
0 1 1 1
2 1 1 1
1
```

```
> cat data2.txt
4
11 7 5 1
15 6 2 2
5 8 5 1
19 4 8 1
132
```

```
> cat data3.txt
4
11 7 5 1
15 6 2 2
5 8 5 1
19 4 8 1
78
```



Ισοζυγισμένα βάρη (0.25+0.25 = 0.5 βαθμοί)

Έχετε στη διάθεσή σας ένα σύνολο από *N* αντικείμενα αριθμημένα 1, 2, ... *N*, όπου το κάθε αντικείμενο έχει από ένα βάρος $w_i = 3^{i-1}$. Με άλλα λόγια, το πρώτο αντικείμενο έχει βάρος 1, το δεύτερο 3, το τρίτο 9 και το *N*-οστό 3^{N-1}. Κάποιος βάζει ένα αντικείμενο με βάρος *W* από τη μία μεριά μιας ζυγαριάς και σας ζητάει να τοποθετήσετε, αν μπορείτε, κάποιο υποσύνολο από τα *N* αντικείμενά σας στη ζυγαριά, πιθανά βάζοντας αντικείμενα και στις δυο μεριές της, έτσι ώστε το συνολικό βάρος να είναι το ίδιο και στις δύο μεριές. Εσείς, φυσικά, αντί να κάνετε το παραπάνω «με το χέρι», γράφετε δύο προγράμματα, ένα σε ML και ένα σε Java, που λύνουν το πρόβλημα.

Η είσοδος των προγραμμάτων σας είναι τα *N* και *W*. Η έξοδος των προγραμμάτων σας είναι οι λίστες των αριθμών των αντικειμένων που πρέπει να μπουν στις δύο πλευρές της ζυγαριάς ώστε να ισορροπήσει, ή μια ειδική τιμή (η δυάδα `([],[])`) αν δεν υπάρχει τρόπος να γίνει αυτό. Αρχικά, η μία μεριά της ζυγαριάς, αυτή που θα πρέπει να αναφέρεται πρώτη στην έξοδό σας, έχει το αντικείμενο με βάρος *W*. Η άλλη είναι κενή. Όπως φαίνεται παρακάτω, το αντικείμενο βάρους δεν είναι μέρος της εξόδου. Μόνο οι αριθμοί των αντικειμένων που πρέπει να τοποθετηθούν σε κάθε μεριά, σε αύξουσα σειρά για κάθε μεριά, αν υπάρχει λύση. Παρατηρήστε ότι μπορεί να μη χρειαστεί να τοποθετήσετε κάποιο αντικείμενο σε κάποια από τις δύο μεριές.

Περιορισμοί: $1 \le N \le 20$, $1 \le W \le 4.200.000.000$, όριο χρόνου εκτέλεσης: 10 seconds, όριο μνήμης: 1.5 GB.

Στην άσκηση αυτή, η είσοδος είναι πολύ απλή και δεν έχει νόημα να βρίσκεται σε αρχείο. Παρακάτω δείχνουμε κάποιες πιθανές κλήσεις των προγραμμάτων σε SML/NJ και σε Java.

Σε SML/NJ

```
- balance 10 2;
val it = ([1],[2]) : int list * int list
- balance 10 5;
val it = ([1,2],[3]) : int list * int list
- balance 13 4;
val it = ([],[1,2]) : int list * int list
- balance 4 42;
val it = ([],[]) : int list * int list
```

Σε Java

```
$ java Balance 10 2
[1] [2]
$ java Balance 10 5
[1,2] [3]
$ java Balance 13 4
[] [1,2]
$ java Balance 4 42
[] []
```

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 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 v110.76 ή σε MLton 20100608 ή σε Objective Caml version 4.01.0. Το σύστημα ηλεκτρονικής υποβολής επιτρέπει να επιλέξετε μεταξύ αυτών των υλοποιήσεων της ML.

- Ο κώδικας των προγραμμάτων σε Java μπορεί να βρίσκεται σε περισσότερα του ενός αρχείου αν θέλετε, αλλά θα πρέπει να μπορεί να μεταγλωττιστεί χωρίς προβλήματα με τον Java compiler με εντολές της μορφής: **`javac Deksamenes.java`** και **`javac Balance.java`**. Η υποβολή σας σε Java μπορεί είτε να είναι ένα μόνο **`.java`** αρχείο ή να αποτελείται από ένα **`.zip`** αρχείο ενός directory το οποίο περιέχει τα **`.java`** αρχεία της υποβολής σας (και μόνο αυτά – μην υποβάλετε **`.class`** αρχεία). Σε κάθε περίπτωση, η υποβολή σας πρέπει να έχει ένα αρχείο με τα ονόματα που φαίνονται παραπάνω σε αυτήν την παράγραφο.

- 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.