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

Εαρινό 2016

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 1

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 9/5/2016, 23:59:59

Ανάστροφα αθροίσματα (0.25+0.25 = 0.5 βαθμοί)

Ο *ανάστροφος* ενός φυσικού αριθμού *N* ορίζεται ως ο φυσικός N_r που προκύπτει από την ανάγνωση του *N* από δεξιά προς τα αριστερά, ξεκινώντας από το πρώτο μη μηδενικό ψηφίο που υπάρχει στο τέλος του. Για παράδειγμα, ο ανάστροφος του 42 είναι ο 24, του 24 είναι ο 42 και ο ανάστροφος του 2400 είναι επίσης ο 42.

Σας δίνεται ένας φυσικός αριθμός *M* ($1 \le M \le 10^{100\ 000}$). Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε C/C++ και ένα σε ML) τα οποία βρίσκουν αν ο *M* μπορεί να προκύψει από το άθροισμα κάποιου φυσικού αριθμού *N* και του ανάστροφού του. (Φυσικά, αυτός ο *N* θα πρέπει να μην έχει μηδενικά στα αριστερά του.) Αν *κάποιος* τέτοιος *N* υπάρχει, το πρόγραμμά σας σε C/C++, MLton ή OCaml θα πρέπει να απλά να τυπώνει τον *N* (ή να επιστρέφει τον *N* ως *συμβολοσειρά* στην SML/NJ), αλλιώς θα πρέπει να τυπώνει 0 (επιστρέφει "0").

Η είσοδος του προγράμματός σας διαβάζεται από ένα αρχείο αποτελούμενο από μία μόνο γραμμή, όπως φαίνεται παρακάτω. Η γραμμή αυτή περιέχει τον αριθμό *M*.

Περιορισμοί: όριο χρόνου εκτέλεσης: 10 seconds, όριο μνήμης: 256 MB.

Παρακάτω δίνονται κάποια παραδείγματα σε C και σε ML.

Σε C/C++, MLton, ή σε OCaml

```
> ./revsum m1.txt
2
> ./revsum m2.txt
10
> ./revsum m3.txt
0
> ./revsum m4.txt
42
```

Σε SML/NJ

```
- revsum "m1.txt";
val it = "2" : string
- revsum "m2.txt";
val it = "10" : string
- revsum "m3.txt";
val it = "0" : string
- revsum "m4.txt";
val it = "42" : string
```

όπου τα τέσσερα αρχεία εισόδου είναι τα εξής(η εντολή **cat** είναι εντολή του Unix):

```
> cat m1.txt
4
> cat m2.txt
11
> cat m3.txt
42
> cat m4.txt
66
```

Στο πρώτο παράδειγμα ο αριθμός 2 (2 + 2 = 4) είναι η μοναδική λύση. Στο δεύτερο παράδειγμα ο αριθμός 10 (10 + 1 = 11) είναι η μοναδική σωστή λύση (ο 01 δεν είναι λύση διότι αρχίζει από 0). Στο τρίτο παράδειγμα είναι εύκολο να επιβεβαιώσετε ότι δεν υπάρχει λύση. Τέλος, το τελευταίο παράδειγμα έχει πολλές λύσεις, π.χ. το 33 το 24 και το 42. Τα προγράμματά σας μπορούν να επιστρέφουν οποιαδήποτε από αυτές. Εμείς, φυσικά, επιλέξαμε να δείξουμε το 42.

Δίκαιες μοιρασιές (0.25+0.25 = 0.5 βαθμοί)

Το πόσο δίκαια επιμερίζονται τα «βάρη» σε μια κοινωνία είναι κάτι σχετικό. Αλλά στα μαθηματικά, το πόσο δίκαια είναι μια μοιρασιά αντικειμένων με κάποιο «βάρος» μπορεί να οριστεί αυστηρά. Μας δίνεται λοιπόν και μια ακολουθία από M αντικείμενα αριθμημένα 1, 2, ... M και ένας θετικός ακέραιος $N \leq M$. Το κάθε αντικείμενο έχει από ένα βάρος $w_1, w_2, \ldots w_M$. Αυτό που θέλουμε είναι να κόψουμε την ακολουθία των αντικειμένων σε N σημεία ($0 = p_0 < p_1 < p_2 < \ldots \leq p_N = M$) τέτοια ώστε το i-οστό κομμάτι να περιλαμβάνει τα αντικείμενα με αριθμούς μεταξύ $p_{i-1}+1$ και p_i τέτοια ώστε το συνολικό βάρος του βαρύτερου κομματιού να είναι όσο το δυνατό μικρότερο. Παρατηρήσετε ότι δε μπορούμε να αλλάξουμε τη σειρά των αντικειμένων.

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε C/C++ και ένα σε ML) τα οποία να διαβάζουν τις παραπάνω παραμέτρους και να επιστρέφουν ως έξοδο μια συμβολοσειρά που δείχνει την πιο δίκαιη μοιρασιά. Στη συμβολοσειρά αυτή τα βάρη θα πρέπει να διαχωρίζονται μεταξύ τους από ένα κενό (space) και τα διαφορετικά κομμάτια από " **|** " (ένα κενό, το χαρακτήρα **|** και άλλο ένα κενό). Αν υπάρχουν περισσότερες από μία λύσεις, το πρόγραμμά σας πρέπει να επιστρέφει αυτήν που ελαχιστοποιεί το βάρος στα πρώτα κομμάτια (αυτά προς τα αριστερά).

Τα στοιχεία εισόδου θα διαβάζονται από ένα αρχείο όπως φαίνεται στα παραδείγματα που ακολουθούν. Η πρώτη γραμμή του αρχείου περιέχει τους δύο ακέραιους M ($1 \leq M \leq 42000$) και N. Η δεύτερη γραμμή περιέχει την ακολουθία με τα M βάρη. Τα βάρη είναι θετικοί ακέραιοι το πολύ μέχρι 10 000 000.

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

Παρακάτω δείχνουμε κάποιες πιθανές κλήσεις των προγραμμάτων σε C/C++/MLton ή σε OCaml:

```
> ./fair_parts objects1.txt
42 42 | 42 42
> ./fair_parts objects2.txt
42 | 42 42 | 42 42
> ./fair_parts objects3.txt
10 20 30 40 50 | 60 70 | 80 90
```

και σε SML/NJ:

```
- fair_parts "objects1.txt";
val it = "42 42 | 42 42" : string
- fair_parts "objects2.txt";
val it = "42 | 42 42 | 42 42" : string
- fair_parts "objects3.txt";
val it = "10 20 30 40 50 | 60 70 | 80 90" : string
```

όπου τα αρχεία με τα δεδομένα εισόδου είναι τα εξής:

```
> cat objects1.txt
4 2
42 42 42 42
> cat objects2.txt
5 3
42 42 42 42 42
> cat objects3.txt
9 3
10 20 30 40 50 60 70 80 90
```

Για το πρώτο αρχείο εισόδου η πιο δίκαιη μοιρασιά είναι μοναδική. Στο δεύτερο παράδειγμα, υπάρχουν τρεις δίκαιες μοιρασιές, αλλά μόνο μία από αυτές (αυτή που φαίνεται παραπάνω) ικανοποιεί τον περιορισμό της εκφώνησης ότι κομμάτι με το ελάχιστο συνολικό βάρος είναι το πρώτο από αριστερά. Το πρόγραμμα σε SML/NJ πρέπει να ορίζει μια συνάρτηση με όνομα **`fair_parts`** η οποία να επιστρέφει μια συμβολοσειρά, όπως φαίνεται παραπάνω.

Περαιτέρω οδηγίες για τις ασκήσεις

- Μπορείτε να δουλέψετε σε ομάδες το πολύ δύο ατόμων, τόσο σε αυτή όσο και στις επόμενες σειρές ασκήσεων. Όμως, έχετε υπ' όψη σας ότι, αν δεν περάσετε το μάθημα φέτος, οι βαθμοί των προγραμματιστικών ασκήσεων κρατούνται μόνο για όσους δεν τις έκαναν σε ομάδα αλλά τις έκαναν μόνοι τους.

- 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. κώδικα ταξινόμησης, 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 C πρέπει να είναι σε ένα αρχείο και να μπορούν να μεταγλωττιστούν χωρίς warnings με gcc (version 4.9.2) με μια εντολή της μορφής (π.χ. για την πρώτη άσκηση):

  ```
  gcc –std=c99 -Wall –Werror -O3 -o revsum yourfile.c
  ```

  (Μπορείτε να υποθέσετε κάποια αντίστοιχη εντολή για C++.)

- Τα προγράμματα σε ML πρέπει επίσης 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 σας επιτρέπει να επιλέξετε μεταξύ αυτών των διαλέκτων της ML.

- Η αποστολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του moodle 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 για την ακριβή διαδικασία υποβολής όταν ανοίξει το σύστημα. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο διότι δε θα γίνουν δεκτά από το σύστημα στο οποίο θα υποβληθούν.