ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΑΘΗΜΑ: ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
ΑΚΑΔ. ΕΤΟΣ: 2015-16
ΔΙΔΑΣΚΩΝ: Χ.ΝΟΜΙΚΟΣ

2η Σειρά Εργαστηριακών Ασκήσεων

Οι απαντήσεις θα πρέπει να υποβληθούν με **turnin**, το αργότερο μέχρι την **Τρίτη 5 Απριλίου 2016**, ώρα **20:00**.

- Για τη συγγραφή των συναρτήσεων συνίσταται να χρησιμοποιήσετε το αρχείο πρότυπο Lab2.hs (που υπάρχει στην ιστοσελίδα του μαθήματος), στο οποίο υπάρχουν έτοιμες οι δηλώσεις των τύπων των συναρτήσεων που θα πρέπει να κατασκευάσετε καθώς και μία ισότητα που ορίζει τις συναρτήσεις ώστε να επιστρέφουν μία προκαθορισμένη τιμή για όλες τις τιμές των ορισμάτων. Για να απαντήσετε σε μία άσκηση μπορείτε να αντικαταστήσετε την παραπάνω ισότητα με τις κατάλληλες ισότητες που ορίζουν την τιμή της συνάρτησης. **Δεν θα πρέπει να τροποποιήσετε το τύπο ούτε το όνομα της συνάρτησης**.

- Μπορείτε να χρησιμοποιήσετε όσες βοηθητικές συναρτήσεις θέλετε, οι οποίες θα καλούνται από τις συναρτήσεις που σας ζητείται να υλοποιήσετε. Σε καμία περίπτωση δεν θα πρέπει να προσθέσετε άλλα ορίσματα στις συναρτήσεις που σας ζητούνται (καθώς αυτό συνεπάγεται αλλαγή του τύπου τους).

- Ο έλεγχος της ορθότητας των απαντήσεων θα γίνει με ημι-αυτόματο τρόπο. Σε καμία περίπτωση δεν θα πρέπει ο βαθμολογητής να χρειάζεται να κάνει παρεμβάσεις στο αρχείο που θα υποβάλετε. Συνεπώς θα πρέπει να λάβετε υπόψη τα παρακάτω:

  1. Κάθε μία από τις συναρτήσεις που σας ζητείται να υλοποιήσετε θα πρέπει να έχει το συγκεκριμένο όνομα και το συγκεκριμένο τύπο που περιγράφεται στην εκφώνηση της αντίστοιχης άσκησης και που υπάρχει στο αρχείο πρότυπο Lab2.hs. **Αν σε κάποια άσκηση το όνομα ή ο τύπος της συνάρτησης δεν συμφωνεί με αυτόν που δίνεται στην εκφώνηση, η άσκηση δεν θα βαθμολογηθεί**.
  2. Το αρχείο που θα παραδώσετε δεν θα πρέπει να περιέχει συντακτικά λάθη. Αν υπάρχουν τμήματα κώδικα που περιέχουν συντακτικά λάθη, τότε θα πρέπει να τα διορθώσετε ή να τα αφαιρέσεται πριν από την παράδοση. **Αν το αρχείο που θα υποβάλετε περιέχει συντακτικά λάθη, τότε ολόκληρη η εργαστηριακή άσκηση θα μηδενιστεί**.
  3. Οι συναρτήσεις θα πρέπει να επιστρέφουν αποτέλεσμα για όλες τις τιμές των ορισμάτων που δίνονται για έλεγχο στο τέλος κάθε άσκησης. Αν κάποιες από τις τιμές που επιστρέφουν οι συναρτήσεις δεν είναι σωστές, αυτό

θα ληφθεί υπόψη στη βαθμολογία, ωστόσο η άσκηση θα βαθμολογηθεί κανονικά. **Αν ωστόσο οι συναρτήσεις δεν επιστρέφουν τιμές για κάποιες από τις τιμές ελέγχου (π.χ. προκαλούν υπερχείλιση στοίβας, ατέρμονο υπολογισμό ή κάποιο σφάλμα χρόνου εκτέλεσης) τότε η αντίστοιχη άσκηση δεν θα βαθμολογηθεί**.

4. Κατα τη διόρθωση των ασκήσεων οι βαθμολογητές δεν θα κάνουν κλήσεις στις βοηθητικές συναρτήσεις που ενδεχομένως θα χρησιμοποιήσετε. Η χρήση των βοηθητικών συναρτήσεων θα πρέπει να γίνεται μέσα από τις συναρτήσεις που σας ζητείται να υλοποιήσετε.

- Μετά το τέλος της εκφώνησης κάθε άσκησης δίνονται τιμές που μπορείτε να χρησιμοποιήσετε για έλεγχο της ορθότητας των συναρτήσεων.

- Για υποβολή με turnin γράψτε (υποθέτοντας ότι έχετε γράψει το πρόγραμμα στο αρχείο Lab2.hs):

turnin Haskell-2@myy401 Lab2.hs

Ασκηση 1.
Τα αποτελέσματα των αγώνων μίας ποδοσφαιρικής ομάδα κατά τη διάρκεια του πρωτάθληματος, μπορούν να αναπαρασταθούν ως μία λίστα από ζεύγη ακεραίων. Το κάθε ζεύγος παριστάνει το αποτέλεσμα ενός αγώνα, όπου το πρώτο στοιχείο του ζεύγους αντιστοιχεί στα τέρματα τα οποία πέτυχε η ομάδα και το δεύτερο στοιχείο στα τέρματα που δέχτηκε από την αντίπαλη ομάδα στο συγκεκριμένο αγώνα.

Με δεδομένη μία λίστα που περιλαμβάνει τα αποτέλεσματα μίας ομάδας στο πρωτάθλημα, θέλουμε να κατασκευάσουμε μία πεντάδα αριθμών που να περιγράφει τα παρακάτω στατιστικά της ομάδας:

- το συνολικό πλήθος αγώνων που έχει παίξει
- τους συνολικούς βαθμούς που έχει κερδίσει, με δεδομένο ότι για κάθε νίκη κερδίζει τρεις βαθμούς και για κάθε ισοπαλία ένα βαθμό
- το συνολικό πλήθος τερμάτων που έχει πετύχει
- το συνολικό πλήθος τερμάτων που έχει δεχτεί
- την διαφορά τερμάτων στο καλύτερο αποτέλεσμα που έχει φέρει. Η διαφορά αυτή θα είναι θετική αν η ομάδα έχει επιτύχει τουλάχιστον μία νίκη, μηδέν αν δεν έχει επιτύχει νίκη αλλά έχει τουλάχιστον μία ισοπαλία και αρνητική αν έχει ηττηθεί σε όλους τους αγώνες.

Γράψτε μία συνάρτηση `statistics` σε Haskell, η οποία θα δέχεται ως όρισμα τη λίστα με τα αποτελέσματα μίας ομάδας και θα επιστρέφει μία πεντάδα ακεραίων με τα στατιστικά στοιχεία που περιγράφονται παραπάνω. Ο τύπος της συνάρτησης θα πρέπει να είναι `[(Int,Int)]->(Int,Int,Int,Int,Int)`. Αν η λίστα είναι κενή τότε θα πρέπει να επιστρέφεται η πεντάδα (0,0,0,0,0). Μπορείτε να υποθέσετε ότι η λίστα είναι πεπερασμένη και ότι κάθε ζεύγος στη λίστα απαρτίζεται από δύο μη αρνητικούς αριθμούς (δηλαδή η λίστα περιέχει μόνο έγκυρα αποτελέσματα). Δεν υπάρχει περιορισμός για το πλήθος τερμάτων που μπορεί να επιτύχει μία ομάδα σε έναν αγώνα, ούτε για το μήκος της λίστας.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> statistics []
(0,0,0,0,0)
Main> statistics [(1,5)]
(1,0,1,5,-4)
Main> statistics [(0,1),(1,3),(1,2)]
(3,0,2,6,-1)
Main> statistics [(0,4),(2,2),(2,3), (0,0)]
(4,2,4,9,0)
Main> statistics [(1,1),(3,0),(0,2),(4,3),(7,1),(3,3),(1,4),(2,1)]
(8,14,21,15,6)
```

Ασκηση 2.
Θέλουμε να σχηματίσουμε μία λίστα με όλες τις λέξεις οι οποίες περιέχονται μέσα σε μία δεδομένη συμβολοσειρά. Ονομάζουμε λέξη ένα μη κενό τμήμα της συμβολοσειράς, το οποίο αποτελείται από συνεχόμενα σύμβολα που είναι όλα λατινικά γράμματα (κεφαλαία ή μικρά) και το οποίο δεν περιέχεται σε ένα ευρύτερο τμήμα με την ίδια ιδιότητα (με άλλα λόγια αριστερά και δεξία μίας λέξης που περιέχεται σε μία συμβολοσειρά δεν βρίσκεται γράμμα του λατινικού αλφαβήτου). Για παράδειγμα στη συμβολοσειρά `"Rockabilly Boogie"` περιέχονται οι λέξεις `"Rockabilly"` και `"Boogie"` , ενώ οι συμβολοσειρές `"Rock"` και `"billy"` δεν αποτελούν λέξεις.

Γράψτε μία συνάρτηση `wordList` σε Haskell, η οποία θα δέχεται ως όρισμα μία συμβολοσειρά και θα επιστρέφει μία λίστα με όλες τις λέξεις που περιέχονται σε αυτή, με τη σειρά εμφάνισής τους. Ο τύπος της συνάρτησης θα πρέπει να είναι `String->[String]`.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> wordList ""
[]
Main> wordList "HAZEL"
["HAZEL"]
Main> wordList "Three Simple Words"
["Three","Simple","Words"]
Main> wordList "s.i.n.g.l.e.s"
["s","i","n","g","l","e","s"]
Main> wordList "BEGIN  ...  END"
["BEGIN","END"]
Main> wordList "Would you like some pizza?"
["Would","you","like","some","pizza"]
Main> wordList "* is a star"
["is","a","star"]
Main> wordList "477392-0900024-4234324324"
[]
Main> wordList "------{middle}------"
["middle"]
Main> wordList "[USER-ID:12Ag$Z?5h-65S], E-mail: iam4got10@cs.uoi.gr"
["USER","ID","Ag","Z","h","S","E","mail","iam","got","cs","uoi","gr"]
```

Ασκηση 3.
Γράψτε μία συνάρτηση move σε Haskell, η οποία θα δέχεται ως ορίσματα μία λίστα s οποιουδήποτε τύπου που υποστηρίζει σύγκριση για ισότητα, ένα στοιχείο x ίδιου τύπου με τα στοιχεία της λίστας και έναν ακέραιο n και θα επιστρέφει τη λίστα που προκύπτει από την s με τον παρακάτω τρόπο:

- αν το x δεν περιέχεται στην s, τότε η επιστρεφόμενη λίστα είναι η s.
- αν το x περιέχεται στην s τότε
  - αν n είναι θετικός αριθμός, τότε η επιστρεφόμενη λίστα προκύπτει από την s με μετακίνηση της πρώτης εμφάνισης του στοιχείου x κατά n θέσεις προς τα δεξιά, ή στο τέλος της λίστας αν η πρώτη εμφάνιση του x στην s ακολουθείται από λιγότερα από n στοιχεία.
  - αν n είναι αρνητικός αριθμός, τότε η επιστρεφόμενη λίστα προκύπτει από την s με μετακίνηση της πρώτης εμφάνισης του στοιχείου x κατά $|n|$ θέσεις προς τα αριστερά, ή στήν αρχή της λίστας αν στην s υπάρχουν λιγότερα από $|n|$ στοιχεία πριν από την πρώτη εμφάνιση του x.
  - αν $n = 0$, τότε η επιστρεφόμενη λίστα προκύπτει από την s με διαγραφή της πρώτης εμφάνισης του στοιχείου x.

Ο τύπος της συνάρτησης θα πρέπει να είναι `Eq u => [u]->u->Int->[u]`. Η συνάρτηση θα πρέπει να επιστρέφει αποτέλεσμα ακόμη και αν το πρώτο της όρισμα είναι μία άπειρη λίστα.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> move [] 3 5
[]
Main> move [1,2,3,4] 0 4
[1,2,3,4]
Main> move [1,2,3,4,3,2,1] 3 3
[1,2,4,3,2,3,1]
Main> move [1,2,3,4,5,6,7,8] 6 7
[1,2,3,4,5,7,8,6]
Main> move "skin" 'k' 0
"sin"
Main> move "factor.bit." '.' (-2)
"fact.orbit."
Main> move "0000.1111.00" '.' (-10)
".00001111.00"
Main> move ["one", "two", "three"] "one" (-2)
["one","two","three"]
Main> move ["one", "two", "three"] "one" 1
["two","one","three"]
```

```
Main> move ["one", "two", "three"] "three" 5
["one","two","three"]
Main> move ["one", "two", "three"] "three" (-2)
["three","one","two"]
Main> move [True] True 0
[]
Main> move [True] True 3
[True]
Main> move [True] True (-1003)
[True]
Main> head (move [1..] 1 1500)
2
Main> head (tail (move [1,5..] 4005 (-1000)))
4005
```

Ασκηση 4.
Με δεδομένη μία πραγματική συνάρτηση f και τρεις πραγματικούς αριθμούς a, b, d, μπορούμε να υπολογίσουμε προσεγγιστικά το ολοκλήρωμα της f στο διάστημα $[a, b]$, χρησιμοποιώντας των παρακάτω αναδρομικό τύπο:

$$\int_a^b f(x)dx \simeq f(\frac{a+b}{2}) \cdot (b-a) \text{ αν } b-a \leq d$$

$$\int_a^b f(x)dx = \int_a^{\frac{a+b}{2}} f(x)dx + \int_{\frac{a+b}{2}}^b f(x)dx \text{ αλλιώς}$$

Γράψτε μία συνάρτηση υψηλότερης τάξης `integral` σε Haskell, η οποία θα δέχεται ως ορίσματα μία πραγματική συνάρτηση f και τρεις πραγματικούς αριθμούς a, b, d, και θα υπολογίζει προσεγγιστικά το ολοκλήρωμα της f στο διάστημα $[a, b]$, χρησιμοποιώντας τον παραπάνω αναδρομικό τύπο. Ο τύπος της `integral` θα πρέπει να είναι `(Double->Double)->Double->Double->Double->Double`. Μπορείτε να υποθέσετε ότι $a \leq b$ και $d > 0$.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές (ελέγξτε την ακρίβεια μέχρι έξι δεκαδικά ψηφία):

```
Main> integral id 0 10 0.001
50.0
Main> integral exp 0 1 0.00001
1.71828182845488
Main> integral sin 0 pi 0.0001
2.00000000076598
Main> integral (^2) (-2) 2 0.0001
5.33333333209157
Main> integral ((1/).(*0.693147180559945)) 1 1024 0.001
9.99999994278443
```

Ασκηση 5.
Γράψτε μία συνάρτηση υψηλότερης τάξης `hof` σε Haskell, η οποία θα δέχεται ως όρισμα μία λίστα ακεραίων και θα επιστρέφει ως αποτέλεσμα τη συνάρτηση από ακέραιους σε ακέραιους, της οποίας η τιμή για το n είναι η θέση της πρώτης εμφάνισης του n στη λίστα ή 0 αν το n δεν εμφανίζεται σε αυτή.

Ο τύπος της `hof` θα πρέπει να είναι `[Integer]->(Integer->Integer)`.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> map (hof []) [1,4,5,8]
[0,0,0,0]
Main> map (hof [1..10]) [1,4,5,8]
[1,4,5,8]
Main> map (hof [10,9..1]) [1,4,5,8]
[10,7,6,3]
Main> map (hof [2,4..10000]) [1,4,5,8]
[0,2,0,4]
Main> map (hof [2,4..4096]) [(-2)^x | x<-[1..20]]
[0,2,0,8,0,32,0,128,0,512,0,2048,0,0,0,0,0,0,0,0]
Main> map (hof [8,32..65536]) [2^x | x<-[1..16]]
[0,0,1,0,2,0,6,0,22,0,86,0,342,0,1366,0]
Main> map (hof [1..]) [2^2^x | x<-[1..3]]
[4,16,256]
Main> map ((hof [2,3,1]).(hof [3,2,1])) [1,2,3]
[2,1,3]
```