ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΑΘΗΜΑ: ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
ΑΚΑΔ. ΕΤΟΣ: 2023-24

1η Σειρά Προγραμματιστικών Ασκήσεων (Haskell)

Οι ασκήσεις αυτές είναι **ατομικές**. Οι απαντήσεις θα πρέπει να υποβληθούν με **turnin**, το αργότερο μέχρι την **Πέμπτη 25 Απριλίου 2024, ώρα 23:59**. Οι δύο σειρές προγραμματιστικών ασκήσεων δίνουν 1.5 μονάδες bonus στον τελικό βαθμό της τελικής και της επαναληπτικής εξέτασης, υπό την προϋπόθεση ότι ο βαθμός του αντίστοιχου γραπτού είναι τουλάχιστον 5. Η βαθμολογία των προγραμματιστικών ασκήσεων δεν μεταφέρεται σε άλλη ακαδημαϊκή χρονιά και δεν λαμβάνεται υπόψη στη βαθμολογία της επί πτυχίω εξέτασης.

Πριν ξεκινήσετε να γράφετε τα προγράμματα που ζητούνται στις ασκήσεις της σειράς αυτής, **διαβάστε πολύ προσεκτικά τις αναλυτικές οδηγίες** που ακολουθούν.

Οδηγίες

- Για να εγκαταστήσετε τη Haskell στον υπολογιστή σας, μπορείτε να κατεβάσετε το διερμηνέα hugs από το σύνδεσμο

  https://www.haskell.org/hugs/pages/downloading-May2006.htm

  Συνοπτικές οδηγίες για τη χρήση του hugs υπάρχουν στις σημειώσεις του μαθήματος.

- Για τη συγγραφή των προγραμμάτων επιτρέπεται να χρησιμοποιήσετε προκαθορισμένες συναρτήσεις και προκαθορισμένους τελεστές **μόνο εφόσον αναφέρονται στις σημειώσεις του μαθήματος**. Δεν επιτρέπεται η χρήση του import.

- Για τη συγγραφή των συναρτήσεων θα πρέπει να χρησιμοποιήσετε το αρχείο πρότυπο Haskell.hs στο οποίο υπάρχουν έτοιμες οι δηλώσεις τύπων των συναρτήσεων που θα πρέπει να κατασκευάσετε καθώς και μια ισότητα που ορίζει τις συναρτήσεις ώστε να επιστρέφουν μια προκαθοριμένη τιμή για όλες τις τιμές των ορισμάτων. Για να απαντήσετε σε μια 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).

- **Αν χρησιμοποιήσετε προκαθορισμένες συναρτήσεις ή τελεστές που δεν αναφέρονται στις σημειώσεις του μαθήματος ή αν χρησιμοποιήσετε το import για να ενσωματώσετε έτοιμο κώδικα, η αντίστοιχη άσκηση δεν θα βαθμολογηθεί.**

- 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. Κάθεμία 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 Haskell.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 αφαιρέσετε πριν από την παράδοση. **Αν το αρχείο που θα υποβάλετε περιέχει συντακτικά λάθη, τότε ολόκληρη η εργαστηριακή άσκηση θα μηδενιστεί**.
  3. 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.

- Για υποβολή με turnin γράψτε:

turnin Haskell@myy401 Haskell.hs

Ασκηση 1.
Σε ένα τυχερό παιχνίδι κάθε Σάββατο γίνεται κλήρωση ενός δεκαψήφιου τυχερού αριθμού μεταξύ του 1000000000 και του 9999999999. Ένας παίκτης ο οποίος έχει στοιχηματίσει πάνω σε έναν αριθμό, κερδίζει ένα ποσό που εξαρτάται από το πλήθος των ψηφίων του αριθμού αυτού τα οποία ταυτίζονται με τα ψηφία του τυχερού αριθμού που βρίσκονται στις αντίστοιχες θέσεις, σύμφωνα με τον παρακάτω πίνακα:

Ψηφία που συμφωνούν	Ποσό που κερδίζει ο παίκτης
10	10000000
9	1000000
8	75000
7	10000
6	1250
5	333
4	45
3	10
2	3
λιγότερο από 2	0

Γράψτε μία συνάρτηση `luckyDigits` σε Haskell η οποία θα δέχεται ως παραμέτρους τον τυχερό αριθμό και τον αριθμό που έχει επιλέξει ένα παίκτης και θα υπολογίζει το ποσό που κερδίζει ο παίκτης. Ο τύπος της συνάρτησης `luckyDigits` θα πρέπει να είναι `Integer->Integer->Int`. Μπορείτε να υποθέσετε ότι τα δύο ορίσματα είναι πάντοτε δεκαψήφιοι αριθμοί. Δεν επιτρέπεται να χρησιμοποιήσετε λίστες.

Ασκηση 2.
Γράψτε μία συνάρτηση `ab` σε Haskell η οποία θα δέχεται ως παράμετρο έναν θετικό ακέραιο αριθμό n και θα επιστρέφει ένα ζεύγος ακεραίων (a, b), τέτοια ώστε $1 \leq a \leq b$, $a \cdot b = n$ και το $b - a$ να είναι το ελάχιστο δυνατό. Ο τύπος της συνάρτησης `ab` θα πρέπει να είναι `Int->(Int,Int)`. Μπορείτε να υποθέσετε ότι n είναι μη αρνητικός ακέραιος αριθμός. Δεν επιτρέπεται να χρησιμοποιήσετε λίστες.

Ασκηση 3.
Σκοπός αυτής της άσκησης είναι η κατασκευή μιας συνάρτησης η οποία θα επιστρέφει το πλήθος των δευτερολέπτων που έχουν περάσει από την αρχή του έτους μεχρι μία συγκεκριμένη χρονική στιγμή μέσα στο ίδιο έτος. Για απλούστευση υποθέτουμε ότι το έτος δεν είναι δίσεκτο.

Μια ημερομηνία περιγράφεται ως ένα ζεύγος ακεραίων αριθμών, όπου το δεύτερο στοιχείο του είναι ένας αριθμός ανάμεσα στο 1 και το 12 που αναπαριστά τον μήνα και το πρώτο στοιχείο του είναι ένας αριθμός ανάμεσα στο 1 και το μέγιστο πλήθος ημερών του μήνα που υποδεικνύει το δεύτερο στοιχείο του.

Η ώρα περιγράφεται ως μία τριάδα ακεραίων αριθμών, όπου το πρώτο στοιχείο της είναι ένας αριθμός ανάμεσα στο 0 και το 23 που αναπαριστά τις ώρες, ενώ το δεύτερο και το τρίτο στοιχείο της είναι αριθμοί ανάμεσα στο 0 και το 59 που αναπαριστούν αντίστοιχα τα λεπτά και τα δευτερόλεπτα.

Γράψτε μία συνάρτηση `seconds` σε Haskell, η οποία με παραμέτρους ένα ζεύγος ακεραίων που αναπαριστά την ημερομηνία και μια τριάδα ακεραίων που αναπαριστά την ώρα, θα επιστρέφει το πλήθος των δευτερολέπτων που έχουν περάσει από την αρχή του έτους μέχρι τη χρονική στιγμή που προσδιορίζουν η δεδομένη ημερομηνία και ώρα. Η συνάρτηση θα πρέπει να επιστρέφει την τιμή -777 αν το πρώτο όρισμά της δεν καθορίζει μία έγκυρη ημερομηνία ή αν το δεύτερο όρισμά της δεν καθορίζει μία έγκυρη ώρα. Ο τύπος της συνάρτησης `seconds` θα πρέπει να είναι `(Int,Int)->(Int,Int,Int)->Int`.

Παρακάτω δίνονται ορισμένες τιμές για έλεγχο:

```
Main> seconds (2,1) (0,0,0)
86400
Main> seconds (1,2) (0,0,0)
2678400
Main> seconds (15,4) (23,47,52)
9071272
```

Ασκηση 4.
Γράψτε μία συνάρτηση `deleteIntI` σε Haskell, η οποία θα δέχεται ως παραμέτρους μία λίστα ακεραίων s και μία λίστα θετικών ακεραίων r ταξινομημένη σε γνησίως αύξουσα τάξη και θα επιστρέφει τη λίστα που προκύπτει από την s αν διαγραφούν τα στοιχεία τα οποία βρίσκονται στις θέσεις που υποδεικνύουν τα στοιχεία της r. Τα στοιχεία της r με τιμή μεγαλύτερη από το μήκος της s θα πρέπει να αγνοούνται. Για παράδειγμα αν $s = [4, 2, 6, 7, 3, 5, 2, 1, 3, 5, 2, 1]$ και $r = [3, 5, 6, 9, 15]$, θα πρέπει να επιστρέφεται η λίστα $[4, 2, 7, 2, 1, 5, 2, 1]$. Ο τύπος της συνάρτησης `deleteIntI` θα πρέπει να είναι `[Int]->[Int]->[Int]`. Μπορείτε να υποθέσετε ότι η τα στοιχεία της r είναι θετικοί αριθμοί και ότι είναι ταξινομημένα σε γνησίως αύξουσα τάξη.

Ασκηση 5.
Γράψτε μία συνάρτηση `smooth` σε Haskell, η οποία θα δέχεται ως παραμέτρους μια λίστα ακεραίων αριθμών s και έναν θετικό ακέραιο αριθμό k και θα επιστρέφει τη λίστα ακεραίων αριθμών, το i-στό στοιχείο της οποίας είναι το ακέραιο μέρος του μέσου όρου των k στοιχείων που βρίσκονται στις θέσεις i, $i+1$, ..., $i+k-1$ της λίστας s. Η επιστρεφόμενη λίστα έχει $|s|-k+1$ στοιχεία, όπου $|s|$ το μήκος της λίστας s, αν $|s| \geq k$, αλλιώς είναι η κενή λίστα.

Για παράδειγμα, αν η s περιέχει 10 στοιχεία και $k = 3$, τότε η επιστρεφόμενη λίστα περιέχει 8 στοιχεία. Το πρώτο στοιχείο της επιστρεφόμενης λίστας είναι το ακέραιο μέρος του μέσου όρου των τριών πρώτων στοιχείων της s. Το δεύτερο στοιχείο της επιστρεφόμενης λίστας είναι το ακέραιο μέρος του μέσου όρου του δεύτερου, του τρίτου και του τέταρτου στοιχείου της s. Το τελευταίο (όγδοο) στοιχείο της επιστρεφόμενης λίστας είναι το ακέραιο μέρος του μέσου όρου του όγδοου, του ένατου και του δέκατου στοιχείου της s.

Ο τύπος της συνάρτησης `smooth` θα πρέπει να είναι `[Int]->Int->[Int]`.

Ασκηση 6.
Ονομάζουμε διαμέριση μίας συμβολοσειράς w κάθε ακολουθία αποτελούμενη από δύο ή περισσότερες μη κενές συμβολοσειρές $z_1, z_2, \ldots, z_k$ ($k \geq 2$), η συνένωση των οποίων είναι η w. Κάθε διαμέριση της w αναπαρίσταται ως μία λίστα συμβολοσειρών. Για παράδειγμα, οι διαμερίσεις της w="crazy"είναι οι παρακάτω:
["c","razy"], ["c","r","azy"], ["cr","azy"], ["c","r","a","zy"], ["cr","a","zy"], ["c","ra","zy"], ["cra","zy"], ["c","r","a","z","y"], ["cr","a","z","y"], ["c","ra","z","y"], ["cra","z","y"], ["c","r","az","y"], ["cr","az","y"], ["c","raz","y"], ["craz","y"].

Γράψτε μία συνάρτηση `partition` σε Haskell, η οποία θα δέχεται ως παράμετρο μία συμβολοσειρά w και θα επιστρέφει μία λίστα με όλες τις διαμερίσεις της w. Η σειρά με την οποία θα εμφανίζονται οι διαμερίσεις μέσα στη λίστα μπορεί να είναι οποιαδήποτε, αρκεί κάθε διαμέριση να εμφανίζεται ακριβώς μία φορά. Ο τύπος της συνάρτησης `partition` θα πρέπει να είναι `String->[[String]]`.

Ασκηση 7.
Γράψτε μία συνάρτηση `move` σε Haskell, η οποία θα δέχεται ως παραμέτρους 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 `move` θα πρέπει να είναι `Eq u => [u]->u->Int->[u]`. Η συνάρτηση θα πρέπει να επιστρέφει αποτέλεσμα ακόμη και στην περίπτωση που το πρώτο της όρισμα είναι μία άπειρη λίστα η οποία περιέχει το στοιχείο x.

Παρακάτω δίνονται ορισμένες τιμές για έλεγχο:

```
Main> move [1,3,5,8,10,12,15] 8 2
[1,3,5,10,12,8,15]
Main> move "factor.bit." '.' (-2)
"fact.orbit."
Main> move [1,3,5,8,10,5,7] 5 0
[1,3,8,10,5,7]
```

Ασκηση 8.
Για κάθε συνάρτηση f από ακέραιους σε ακέραιους ορίζουμε την αντίστροφή της f^{-1} με τον παρακάτω τρόπο:

$$f^{-1}(n) = \min\{k \in \mathbb{N} \mid k \geq 0 \text{ και } f(k) = n\}$$

(δηλαδή το $f^{-1}(n)$ είναι ο ελάχιστος μη αρνητικός ακέραιος αριθμός k για τον οποίο ισχύει $f(k) = n$).

Γράψτε μία συνάρτηση υψηλότερης τάξης `inverse` σε Haskell, η οποία θα δέχεται ως παράμετρο μία συνάρτηση f από ακέραιους σε ακέραιους και θα επιστρέφει τη συνάρτηση f^{-1}.

Ο τύπος της συνάρτησης `inverse` θα πρέπει να είναι `(Int->Int)->(Int->Int)`.

Ασκηση 9.
Γράψτε μία συνάρτηση υψηλότερης τάξης `sumfab` σε Haskell, η οποία με παραμέτρους μία συνάρτηση f τύπου `Int->Int->Int->Int` και δύο ακέραιους a και b θα επιστρέφει το άθροισμα

$$\sum_{k=a}^{b} f(a, k, b)$$

Ο τύπος της συνάρτησης `sumfab` θα πρέπει να είναι `(Int->Int->Int->Int)->Int->Int->Int`.

Δεν επιτρέπεται να χρησιμοποιήσετε λίστες.

Ασκηση 10.
Γράψτε μία συνάρτηση υψηλότερης τάξης `hof` σε Haskell, η οποία θα δέχεται ως όρισμα μία μη κενή λίστα συναρτήσεων από ακεραίους σε ακεραίους $[f_1, f_2, \ldots, f_k]$ και θα επιστρέφει ως αποτέλεσμα τη συνάρτηση f από ακεραίους σε ακεραίους, η οποία ορίζεται με τον παρακάτω τρόπο:

$$f(n) = \sum_{i=1}^{k} \lfloor \frac{f_i(n-i+1)}{2^{i-1}} \rfloor = f_1(n) + \lfloor \frac{f_2(n-1)}{2} \rfloor + \lfloor \frac{f_3(n-2)}{4} \rfloor + \ldots + \lfloor \frac{f_k(n-k+1)}{2^{k-1}} \rfloor$$

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