

3η Σειρά Εργαστηριακών Ασκήσεων

Οι εργαστηριακές ασκήσεις είναι ατομικές. Οι απαντήσεις θα πρέπει να υποβληθούν με **turnin**, το αργότερο μέχρι την **Τρίτη 12 Απριλίου 2022, ώρα 23:59**.

Οι ώρες οι οποίες έχουν δεσμευτεί για το εργαστήριο του μαθήματος είναι την Παρασκευή 10-12πμ. Η παρουσία στο εργαστήριο τις παραπάνω ώρες δεν είναι υποχρεωτική. Μπορείτε να έρχετε στο εργαστήριο τις ώρες αυτές για όποια βοήθεια χρειάζεστε σχετικά με την εκπόνηση των εργαστηριακών ασκήσεων και γενικότερα τον προγραμματισμό στη γλώσσα Haskell, καθώς και για την επίλυση προβλημάτων που παρουσιάζονται κατά τη συγγραφή των προγραμμάτων στο πλαίσιο των εργαστηριακών ασκήσεων. Για αντίστοιχα προβλήματα ή απορίες που θα προκύψουν στο διάστημα από την περάτωση του εργαστηρίου μέχρι την υποβολή της εργασίας μπορείτε να επικοινωνήσετε με την κ. Βίκυ Σταμάτη την Τρίτη 10:00-12:00 είτε δια ζώσης (Γραφείο B11) είτε μέσω ηλεκτρονικού ταχυδρομείου (vstamati@uoi.gr). Θα απαντηθούν μόνο ηλεκτρονικά μηνύματα που έχουν σταλεί από τον ιδρυματικό σας λογαριασμό.

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

Οδηγίες

- 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 `Lab3.hs` (που υπάρχει στην ιστοσελίδα του μαθήματος), στο οποίο υπάρχουν έτοιμες οι δηλώσεις 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).
- 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. 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 Lab3.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. 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:

turnin Haskell-3@myy401 Lab3.hs

Άσκηση 1.

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 δεξιά 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"]
```

Άσκηση 2.

Ένα ρομπότ μπορεί να κινείται σε ένα επίπεδο, το οποίο είναι χωρισμένο σε τετράγωνους τομείς. Κάθε τομέας περιγράφεται από ένα ζεύγος ακεραίων αριθμών που καθορίζει τις συντεταγμένες του. Το ρομπότ για να κινηθεί από έναν τομέα αφετηρία (x_1, y_1) σε έναν τομέα προορισμό (x_2, y_2) , αρχικά κινείται διαγώνια προς την κατάλληλη κατεύθυνση, μέχρι κάποια από τις δύο συνιστώσες του να γίνει ίση με την αντίστοιχη του τομέα προορισμού. Στη συνέχεια, κινείται οριζόντια ή κάθετα μέχρι να φτάσει στον τομέα προορισμό. Για παράδειγμα, για να κινηθεί από τον τομέα $(-2, 3)$ στον τομέα $(4, 5)$, το ρομπότ θα περάσει από τους τομείς $(-1, 4)$, $(0, 5)$, $(1, 5)$, $(2, 5)$ και $(3, 5)$, ενώ για να κινηθεί από τον τομέα $(3, 8)$ στον τομέα $(9, 0)$, θα περάσει από τους τομείς $(4, 7)$, $(5, 6)$, $(6, 5)$, $(7, 4)$, $(8, 3)$, $(9, 2)$ και $(9, 1)$.

Το ρομπότ καθημερινά εκτελεί μία σειρά από εργασίες που πρέπει να γίνουν με προκαθορισμένη σειρά σε κάποιους τομείς $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. Το ρομπότ ξεκινώντας από την αρχική του θέση που είναι ο τομέας $(0, 0)$, θα πρέπει να μεταβεί πρώτα στον τομέα (x_1, y_1) , να εκτελέσει εκεί την πρώτη εργασία, στη συνέχεια να μεταβεί στον τομέα (x_2, y_2) και να εκτελέσει εκεί τη δεύτερη εργασία και να συνεχίσει με τον ίδιο τρόπο, μέχρι να ολοκληρώσει και την τελευταία εργασία του. Τέλος από τον τομέα (x_n, y_n) πρέπει να επιστρέψει στην αρχική του θέση $(0, 0)$.

Γράψτε μία συνάρτηση `trace` σε Haskell, η οποία θα δέχεται ως όρισμα μία λίστα από ζεύγη ακεραίων, η οποία περιέχει τις συντεταγμένες των τομέων στις οποίες θα πρέπει να εκτελεστούν οι εργασίες από το ρομπότ και θα επιστρέφει μία λίστα του ίδιου τύπου, η οποία θα περιέχει όλους τους τομείς που θα επισκεφτεί το ρομπότ ξεκινώντας από τη θέση $(0, 0)$ έτσι ώστε να εκτελέσει τις απαραίτητες εργασίες σε όλους τους τομείς της λίστας και να επιστρέψει στη θέση $(0, 0)$. Μπορείτε να υποθέσετε ότι η λίστα δεν περιέχει το στοιχείο $(0, 0)$ και ότι διαδοχικές εργασίες εκτελούνται σε διαφορετικούς τομείς. Ο τύπος της συνάρτησης θα πρέπει να είναι `[(Int,Int)]->[(Int,Int)]`

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> trace [(3,8)]
[(0,0),(1,1),(2,2),(3,3),(3,4),(3,5),(3,6),(3,7),(3,8),(2,7),(1,6),(0,5),(0,4),(0,3),
(0,2),(0,1),(0,0)]
```

```
Main> trace [(5,5),(-5,5)]
[(0,0),(1,1),(2,2),(3,3),(4,4),(5,5),(4,5),(3,5),(2,5),(1,5),(0,5),(-1,5),(-2,5),
(-3,5),(-4,5),(-5,5),(-4,4),(-3,3),(-2,2),(-1,1),(0,0)]
```

```
Main> trace [(5,3),(-2,-6),(-1,1),(3,-7)]
[(0,0),(1,1),(2,2),(3,3),(4,3),(5,3),(4,2),(3,1),(2,0),(1,-1),(0,-2),(-1,-3),(-2,-4),
(-2,-5),(-2,-6),(-1,-5),(-1,-4),(-1,-3),(-1,-2),(-1,-1),(-1,0),(-1,1),(0,0),(1,-1),
(2,-2),(3,-3),(3,-4),(3,-5),(3,-6),(3,-7),(2,-6),(1,-5),(0,-4),(0,-3),(0,-2),(0,-1),
0,0)]
```

```
Main> trace [(i,2*(-1)^i) | i <- [1..12]]
[(0,0),(1,-1),(1,-2),(2,-1),(2,0),(2,1),(2,2),(3,1),(3,0),(3,-1),(3,-2),(4,-1),(4,0),
(4,1),(4,2),(5,1),(5,0),(5,-1),(5,-2),(6,-1),(6,0),(6,1),(6,2),(7,1),(7,0),(7,-1),
(7,-2),(8,-1),(8,0),(8,1),(8,2),(9,1),(9,0),(9,-1),(9,-2),(10,-1),(10,0),(10,1),(10,2),
(11,1),(11,0),(11,-1),(11,-2),(12,-1),(12,0),(12,1),(12,2),(11,1),(10,0),(9,0),(8,0),
(7,0),(6,0),(5,0),(4,0),(3,0),(2,0),(1,0),(0,0)]
```