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

1η Σειρά Εργαστηριακών Ασκήσεων

Οι απαντήσεις θα πρέπει να το υποβληθούν με **turnin**, το αργότερο μέχρι την **Τετάρτη 9 Μαρτίου 2016**, ώρα **20:00**.

- Για μεγαλύτερη ευκολία στην εκπόνηση αυτής και της επόμενης εργαστηριακής άσκησης σας συνιστώ:

  1. να δημιουργήσετε έναν κατάλογο Haskell κάτω από το home directory σας.
  2. να χρησιμοποιήτε αυτόν τον κατάλογο για αποθήκευση των αρχείων με τα προγράμματα Haskell που θα γράψετε.
  3. να μεταβαίνετε σε αυτόν τον κατάλογο πριν εκτελέσετε τον διερμηνέα hugs της Haskell.

- Αν θέλετε να χρησιμοποιήσετε τη Haskell στο δικό σας υπολογιστή, μπορείτε να κατεβάσετε το διερμηνέα hugs από το σύνδεσμο

  https://www.haskell.org/hugs/

  Συνοπτικές οδηγίες για τη χρήση του hugs υπάρχουν στις σημειώσεις.

- Πριν ξεκινήσετε να γράφετε τα προγράμματα που ζητούνται στις παρακάτω ασκήσεις, θα ήταν χρήσιμο να γράψετε σε ένα αρχείο ορισμένες από τις συναρτήσεις των σημειώσεων, να φορτώσετε το αρχείο στον hugs και να αποτιμήσετε παραστάσεις που χρησιμοποιούν τις συναρτήσεις αυτές, έτσι ώστε να εξοικειωθείτε με την γλώσσα Haskell και το διερμηνέα της.

- Για τη συγγραφή των συναρτήσεων συνίσταται να χρησιμοποιήσετε το αρχείο πρότυπο Lab1.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 την τιμή -2016 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 Lab1.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 Lab1.hs):

turnin Haskell-1@myy401 Lab1.hs

Ασκηση 1.
Μία μεγάλη αλυσίδα καταστημάτων εφαρμόζει μία πολιτική προσφορών και εκπτώσεων έτσι ώστε να αυξήσει τις πωλήσεις των προϊόντων της. Σύμφωνα με την πολιτική των πωλήσεων της αλυσίδας αυτής για την αγορά κάθε προϊόντος:

- στις πρώτες πέντε συσκευασίες του προϊόντος η μία είναι δωρεάν
- στις επόμενες τέσσερεις συσκευασίες του προϊόντος η μία είναι δωρεάν
- σε κάθε επιπλέον τρεις συσκευασίες του προϊόντος η μία είναι δωρεάν.
- αν το συνολικό κόστος που προκύπτει από την αγορά του προϊόντος είναι μεγαλύτερο από 100 ΕΥΡΩ τότε υπάρχει επιπλέον έκπτωση 10%.

Γράψτε μία συνάρτηση `cost` σε Haskell η οποία θα δέχεται ως ορίσματα το πλήθος των συσκευασιών ενός προϊόντος που αγοράζει κάποιος πελάτης και την τιμή της μίας συσκευασίας του προϊόντος και θα επιστρέφει το συνολικό κόστος που προκύπτει με βάση την πολιτική πωλήσεων της αλυσίδας που περιγράφεται παραπάνω. Ο τύπος της συνάρτησης θα πρέπει να είναι `Int->Float->Float`. Αν κάποιο από τα δύο ορίσματα δέν είναι θετικός αριθμός, τότε η συνάρτηση cost θα πρέπει να επιστρέφει την τιμή 0.

Για τη μετατροπή ακέραιου σε πραγματικό χρησιμοποιήστε τη συνάρτηση `fromIntegral`.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> cost 3 32.0
96.0
Main> cost 4 32.0
115.2
Main> cost 5 32.0
115.2
Main> cost 8 0.15
1.05
Main> cost 9 0.15
1.05
Main> cost 35 4.25
95.625
Main> cost 36 4
100.0
Main> cost 37 4
93.6
Main> cost (-5) 15.55
0.0
Main> cost 10 (-3.77)
0.0
```

Ασκηση 2.
Μία εταιρία κινητής τηλεφωνίας χρεώνει κάθε κλήση διάρκειας μέχρι 3 λεπτά προς οποιονδήποτε αριθμό με 0.58 ΕΥΡΩ και αν η διάρκεια της κλήσης υπερβεί τα 3 λεπτά, τότε ο επιπλέον χρόνος χρεώνεται με 0.003 ΕΥΡΩ το δευτερόλεπτο. Κλήσεις με μηδενική διάρκεια δεν χρεώνονται.

Γράψτε μία συνάρτηση `call` σε Haskell η οποία θα δέχεται ως ορίσματα τις ώρες έναρξης και λήξης μίας κλήσης και θα υπολογίζει τη συνολική χρέωση για την κλήση. Η ώρα παριστάνεται ως μία τριάδα ακεραίων (για παράδειγμα η ώρα 15:18:31 παριστάνεται ως $(15, 18, 31)$). Ο τύπος της συνάρτησης θα πρέπει να είναι `(Int,Int,Int)->(Int,Int,Int)->Float`. Μπορείτε να υποθέσετε ότι τα δύο ορίσματα είναι πάντοτε έγκυρα (δηλαδή αντιστοιχούν σε σωστές ένδειξεις ώρας) και επιπλέον ότι η διάρκεια μίας κλήσης είναι μικρότερη από 24 ώρες.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> call (17,23,15) (17,23,15)
0.0
Main> call (8,12,3) (8,12,58)
0.58
Main> call (12,0,35) (12,1,24)
0.58
Main> call (16,58,35) (17,0,0)
0.58
Main> call (23,59,42) (0,1,40)
0.58
Main> call (14,32,8) (14,35,17)
0.607
Main> call (14,57,4) (15,0,23)
0.637
Main> call (23,59,59) (0,2,59)
0.58
Main> call (3,15,22) (11,55,8)
93.598
Main> call (19,43,48) (1,5,7)
57.877
```

Ασκηση 3.
Γράψτε μία συνάρτηση `pow` σε Haskell η οποία θα δέχεται ως όρισμα ένα θετικό ακέραιο n και θα επιστρέφει την πλησιέστερη στον n δύναμη του 2. Αν ο αριθμός απέχει το ίδιο από δύο δυνάμεις του δύο, τότε θα πρέπει να επιλέγεται η μεγαλύτερη από τις δύο δυνάμεις. Ο τύπος της συνάρτησης θα πρέπει να είναι `Integer->Integer`. Μπορείτε να υποθέσετε ότι το όρισμα είναι πάντοτε θετικός ακέραιος.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> pow 1
1
Main> pow 2
2
Main> pow 3
4
Main> pow 6
8
Main> pow 23
16
Main> pow 24
32
Main> pow 100
128
Main> pow 750
512
Main> pow 15000
16384
Main> pow 1000000000
1073741824
```

Ασκηση 4.
Μπορούμε να μετατρέψουμε έναν δεδομένο θετικό ακέραιο αριθμό σε έναν μονοψήφιο αριθμό με την παρακάτω διαδιακασία: ενώσο ο αριθμός δεν είναι μονοψήφιος τον αντικαθιστούμε με το γινόμενο των μη μηδενικών ψηφίων του. Σε έναν αριθμό που έχει μόνο ένα μη μηδενικό ψηφίο, το αποτέλεσμα ισούται με τον μονοψήφιο που καθορίζεται από το ψηφίο αυτό. Π.χ. 1978→504→20→2.

Γράψτε μία συνάρτηση `num2dig` σε Haskell, η οποία θα δέχεται ως όρισμα έναν ακέραιο αριθμό n και θα τον μετατρέπει σε μονοψήφιο εφαρμόζοντας την παραπάνω διαδικασία. Ο τύπος της συνάρτησης θα πρέπει να είναι `Integer->Integer`. Μπορείτε να υποθέσετε ότι ο n είναι μη αρνητικός ακέραιος αριθμός.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> num2dig 7
7
Main> num2dig 32
6
Main> num2dig 35
5
Main> num2dig 39
4
Main> num2dig 58
4
Main> num2dig 7235
2
Main> num2dig 87251
3
Main> num2dig 25522525
1
Main> num2dig (11^15)
2
Main> num2dig (13^7128)
8
```

Ασκηση 5.
Γράψτε μία συνάρτηση `ijk` σε Haskell η οποία θα δέχεται ως όρισμα έναν θετικό ακέραιο αριθμό n και θα επιστρέφει μία τριάδα ακεραίων (i, j, k), τέτοια ώστε $1 \leq i \leq j \leq k$, $ijk = n$ και το $k - j$ να είναι το ελάχιστο δυνατό. Αν υπάρχουν περισσότερες από μία τριάδες που ελαχιστοποιούν το $k - j$, να επιστρέφεται αυτή στην οποία το k είναι το ελάχιστο. Ο τύπος της συνάρτησης θα πρέπει να είναι `Int->(Int,Int,Int)`. Μπορείτε να υποθέσετε ότι n είναι μη αρνητικός ακέραιος αριθμός.

Για έλεγχο χρησιμοποιήστε τις παρακάτω τιμές:

```
Main> ijk 23
(1,1,23)
Main> ijk 24
(2,3,4)
Main> ijk 25
(1,5,5)
Main> ijk 27
(3,3,3)
Main> ijk 64
(4,4,4)
Main> ijk 72
(2,6,6)
Main> ijk 108
(3,6,6)
Main> ijk 150
(5,5,6)
Main> ijk 210
(5,6,7)
Main> ijk 280
(5,7,8)
```