Ημερομηνία Ανάρτησης: 16/1/2016
Ημερομηνία Παράδοσης: 31/1/2016, 23:59μμ
Αρχές Γλωσσών Προγραμματισμού

1. (20%) Ορίστε σε Haskell τη συνάρτηση `sssum f a b c` η οποία δεδομένης μιας συνάρτησης `f` με τύπο `Int->Int->Int->Int`, και τριών φυσικών αριθμών `a`, `b`, `c`, υπολογίζει την παράσταση

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} f\ i\ j\ k$$

   Η υλοποίησή σας θα πρέπει να γίνει χωρίς τη χρήση list comprehensions.

2. (20%) Μία λίστα ονομάζεται 'τυχερή' αν περιέχει μόνο τα ψηφία 2,3,5 και 7. Για παράδειγμα, `[2,3,5]`, `[7,5,3]`, `[2,2,2,3]` και `[5,3,2,2,7]` είναι τυχερές λίστες και οι `[2,2,1]`, `[4,7,4,7,4]` και `[7,5,4,2,2,3]` δεν είναι. Μια τυχερή λίστα ονομάζεται 'cool' αν περιέχει τουλάχιστον 2 δυάρια στη σειρά (όπως η `[3,5,2,2,5]`), τουλάχιστον 3 τριάρια στη σειρά (όπως η `[3,5,3,3,3,3]`), τουλάχιστον 5 πεντάρια στη σειρά (όπως η `[5,5,5,5,5,2,3,7]`), τουλάχιστον 7 εφτάρια στη σειρά (όπως η `[7,7,7,7,7,7,7,7,7]`), ή οποιονδήποτε συνδυασμό από τους παραπάνω (όπως η `[3,3,3,5,2,2,2,2,7]`). Ορίστε σε Haskell συνάρτηση `cool xs` η οποία δεδομένης μιας λίστας που περιέχει τα ψηφία 2, 3, 5, 7 και μηδενικά, τυπώνει αν μπορούν ή όχι τα μηδενικά στη λίστα να αντικατασταθούν με τα ψηφία 2, 3, 5 και 7 και να δώσουν μία cool λίστα (κάθε μηδενικό μπορεί να αντικατασταθεί μόνο με ένα ψηφίο). Για παράδειγμα, `cool [2,3,3,0,5,7,5,7] = True` ενώ `cool [5,7,0,5,0,7,5,7,0,3] = False`.

3. (20%) Έστω μια λίστα η οποία περιέχει όλους τους αριθμούς από το 1 μέχρι το n, χωρίς επαναλήψεις και όχι υποχρεωτικά στη σειρά. Υποθέτουμε ότι τη λίστα αυτή αρχικά δεν τη γνωρίζουμε. Γνωρίζουμε όμως πέντε λίστες που έχουν προκύψει από την άγνωστη λίστα με τον ακόλουθο τρόπο. Η πρώτη λίστα έχει προκύψει από την άγνωστη λίστα με μετακίνηση ενός από τους αριθμούς σε μια άλλη θέση. Η δεύτερη λίστα έχει προκύψει από τη μετακίνηση ενός *διαφορετικού* αριθμού από την *αρχική* λίστα σε μια άλλη θέση, κλπ. Γράψτε συνάρτηση `findlist` σε Haskell η οποία δεδομένης μιας λίστας που έχει για στοιχεία πέντε λίστες, επιστρέφει ως αποτέλεσμα την αρχική λίστα.

4. (20%) Το ίδιο με το 4ο πρόβλημα της πρώτης εργασίας. Ορίστε σε Haskell τη συνάρτηση `doses n m ls ws` η οποία δεδομένων `n`, `m` και λιστών `ls` και `ws` επιστρέφει τον αριθμό των δόσεων που ενδεχομένως θα χρειαστεί η παρέα. Για παράδειγμα, η κλήση: `doses 3 4 [(1,1,1),(1,3,4),(3,1,3),(2,1,5),(2,2,7)] [(1,3),(2,8)]` θα επιστρέψει `3`.

5. (20%) Το ίδιο με το 5ο πρόβλημα της πρώτης εργασίας. Ορίστε σε Haskell τη συνάρτηση `maxdiff ls ms` η οποία δεδομένων δύο λιστών που περιέχουν η μεν `ls` τη διαδρομή μαζί με τα όρια ταχύτητας και η δε `ms` το ταξίδι του Γιώργου, επιστρέφει τη μέγιστη υπέρβαση του ορίου ταχύτητας που έγινε κατά τη διάρκεια του ταξιδιού. Αν το όριο δεν ξεπεράστηκε ποτέ, θα πρέπει να επιστρέφεται η τιμή 0. Για παράδειγμα, η κλήση: `maxdiff [(40,75),(50,35),(10,45)] [(40,76),(20,30),(40,40)]` θα επιστρέψει `5`.

Παράδοση Ασκήσεων: Η παράδοση πρέπει να γίνει μέχρι τις 23:59μμ, την 31/1/2016. Θα δημιουργήσετε **ένα** αρχείο το οποίο θα περιέχει τις λύσεις όλων των ασκήσεων το οποίο θα στείλετε με *email* στο `antru@di.uoa.gr` (με κοινοποίηση στο `prondo@di.uoa.gr`) μέχρι την παραπάνω ημερομηνία. **Δεν θα υπάρξει παράταση στην παράδοση των ασκήσεων**. Τα ονόματα των συναρτήσεων που θα χρησιμοποιήσετε στα προγράμματά σας πρέπει να είναι **ακριβώς** τα ίδια με αυτά που καθορίζονται από την παραπάνω εκφώνηση. Καθυστερημένες ασκήσεις δεν θα βαθμολογηθούν.

Σημείωση: Για να μπορέσει κάποιος να λάβει μέρος στην τελική εξέταση του μαθήματος, θα πρέπει να έχει παραδώσει τις δύο πρώτες εργασίες (Prolog και Haskell).