Programowanie funkcyjne

Kolokwium
20 stycznia 2016

1. a) (2) Proszę zdefiniować funkcję `member x l`, która sprawdza, czy element `x` jest w liście `l`.

   b) (2) Proszę napisać funkcję `delete x l`, która skasuje wszystkie wystąpienia elementu `x` w liście `l`.

   c) (2) Proszę napisać funkcję `exists pred l` dla jednoargumentowego predykatu `pred`, której wartość to `False`, jeżeli wszystkie elementy `x` w liście `l` nie spełniają predykat `pred`, a `True` w przeciwnym przepadku.
   Uwaga: Należy używać odpowiedniego kombinatora!

2. a) (2) Proszkę zdefiniować funkcję `cut l`, która skasuje pierwszy i ostatni element z listy `l`. Jeżeli `l` ma mniej niż dwa elementy, wartością ma być pusta lista.

   ```
   Przykład: > cut [4,3,2,0,1,2]
             [3,2,0,1]
   ```

   b) (2) Proszę zdefiniować funkcję `palindrom l`, która sprawdza, czy lista `l` zawiera palindrom.

   ```
   Przykład: > palindrom [1,1,2,4,4,1]
             False
             > palindrom [1,2,3,2,1]
             True
   ```

   Uwaga: Można używać funkcje `reverse` oraz `last`.

3. Niech będzie dany następujący typ drzew.

   ```
   data Tree a b = Leaf a | Node a b (Tree a b) (Tree a b)
   ```

   a) (3) Proszę zdefiniować funkcję `sumTree t`, która obliczy sumę elementów typu `a` w drzewie `t`.

   b) (3) Proszę zdefiniować funkcję `preTree t`, której wartością jest lista elementów drzewa `t` w porządku prefiksowym.

   c) (3) Proszę zdefiniować funkcję `mapb f t`, która zastosuje funkcję `f` do wszystkich elementów typu `b`.

   Należy też podac typy zdefiniowanych funkcji.

4. a) (3) Proszę obliczyć typ wyrażenia $\backslash f \to fx$.

   b) (3) Proszę obliczyć typ wyrażenia $2 * (x + y)$.
   Uwaga: Można wykorzystać `2 : Int`, `(+) : Int -> Int -> Int` oraz `(*) : Int -> Int -> Int`.