Lista 6

Algebraiczne typy danych i ewaluacja leniwa

W poniższych zadaniach **dopuszczalne jest** wykorzystanie funkcji wbudowanych obliczających **długość listy**, **odwracających listę** oraz **łączących dwie listy**, o ile **nie wpływają one na drastyczne pogorszenie złożoności obliczeniowej**.

Każde zadanie, poza implementacją funkcji, musi posiadać **kompletny zestaw testów**.

Do wykonania zadań należy wykorzystać mechanizmy poznane na wykładach nr 4 i 5.

1) Zdefiniuj:
   a. Typ *Expression* reprezentujący wyrażenia algebraiczne w postaci: (Scala) (5 pkt.)
      - Elementu Val przechowującego liczbę rzeczywistą,
      - Elementów – znaczników Sum, Diff, Prod, Div – reprezentujących podstawowe operacje algebraiczne.

   b. Zdefiniuj funkcję *evaluate* dokonującą wyliczenia wartości wyrażenia w postaci postfiksowej – jako listy elementów typu Expression. Błędy obsługiwać za pomocą typu opcjonalnego. Ewaluację przeprowadzić z wykorzystaniem stosu – listy. (Scala) (20 pkt.)

      Przykładowo:

      evaluate(List(Val(1.0), Val(3.5), Sum)) == Some(4.5)
      evaluate(List(Val(0.0), Val(1.0), Div)) == None

2) Wykonaj następujące kroki:

   a. Zdefiniuj typ *lazyBinaryTree* reprezentujący leniwe drzewo binarne. Wykorzystaj abstrakcję funkcyjną. (OCaml) (5 pkt.)

   b. Napisz funkcję *treeFoldL* wykonującą operację fold left na leniwym drzewie binarnym. Drzewo powinno być przechodzone w kolejności inorder. (OCaml) (20 pkt.)

      treeFoldL: (‘a -> ‘b -> ‘a) -> ‘a -> ‘b lazyBinaryTree -> ‘a