Lista 5

Ewaluacja gorliwa i leniwa. Listy leniwe. Przekazywanie argumentów do funkcji

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 wykładzie nr 5.

1) Zdefiniuj:
   a. typ **leniwego drzewa trójkowego** tj. rozszerzenia drzewa binarnego na posiadanie trójki dzieci. W definicji typu wykorzystaj **zmienne leniwe**. (OCaml i Scala) (10 pkt.)

   b. Funkcję *traverse* przyjmującą: funkcję *order* przyjmującą węzeł drzewa i ustalającą dla niego **kolejność przechodzenia** jego dzieci oraz drzewo trójkowe *t*.
   Traverse ma za zadanie przechodzić po drzewie trójkowym na podstawie porządków ustalanych przez funkcję *order* zbierając elementy drzewa do **listy leniwej**.
   W **OCamlu** zaimplementować listę leniwą z wykorzystaniem **abstrakcji funkcyjnej**.
   W **Scali** wykorzystać **typ wbudowany**. (OCaml i Scala) (30 pkt.)

   UWAGA! Dokładny sposób reprezentacji kolejności wykorzystywany przez funkcję *order* nie jest narzucony z góry – zależy od autora. Należy pamiętać jednak, że funkcja *order* może (i zapewne powinna) zwracać **różne kolejności** dla różnych węzłów w drzewie!

2) Zdefiniuj funkcję *skipTakeL*, która dla danej listy leniwej zwraca nową listę leniwą zbudowaną z pierwszego elementu, elementu leżącego w odległości 2 od poprzedniego, z elementu leżącego w odległości 3 od poprzedniego itd. (Scala) (10 pkt.)

   Przykładowo:
   $[x_1; x_2; x_3; x_4; x_5; x_6; x_7; ...]$ --------> $[x_1; x_3; x_6; x_{10}; x_{15}; ...]$