Programowanie funkcyjne — kolokwium nr 1, 8.12.2021

Instrukcja: Każde zadanie należy przesłać na Pegaza w oddzielnym pliku: zadanie1.hs, zadanie2.hs i zadanie3.hs. Plików nie należy zipować. Rozwiązania muszą się poprawnie kompilować. W rozwiązaniach nie można korzystać z modułów innych niż standardowe; niedozwolone jest użycie polecenia import. Rozwiązania nie spełniające powyższych wymogów nie będą oceniane. Punktacja: 10 punktów za każde zadanie.

Zadanie 1. Rozpatrzmy następujący typ reprezentujący graf skierowany:

`type DirectedGraph = ([Int], Int -> Int -> Bool).`

Pierwszy element pary to zbiór wierzchołków V, drugi to funkcja f taka, że dla dowolnych $x, y \in V$ mamy, że $f\ x\ y = \text{True} \iff xy$ jest (skierowaną) krawędzią z x do y. Napisać funkcję

`atDistance :: DirectedGraph -> Int -> Int -> [Int],`

która dla wywołania `atDistance` $g\ d\ v$ zwraca listę (być może z powtórzeniami) wszystkich wierzchołków, które w grafie g są osiągalne z v drogą długości dokładnie d. Dla przypomnienia: w drodze wierzchołki mogą się powtarzać, zaś długość drogi to liczba jej krawędzi.

Zadanie 2. Napisać funkcję o sygnaturze

`wIluListach :: Int -> [[Int]] -> [Int],`

która dla liczby całkowitej dodatniej n oraz listy $[l_1, l_2, ..., l_k]$ list liczb naturalnych z przedziału $\{1, ..., n\}$ sprawdza, w jak wielu listach $l_1, l_2, ..., l_k$ występują kolejne liczby $1, ..., n$. Inaczej mówiąc, funkcja ma zwrócić listę, której pierwszym elementem jest liczba list, w których występuje liczba 1, drugim — liczba list, w których występuje 2 itd. Przykładowo:

`wIluListach 7 [[1,2,3],[3,4,5]]` ma zwrócić `[1,1,2,1,1,0,0]`, ponieważ liczba 1 występuje w jednej liście, podobnie jak 2, 4 i 5; liczba 3 występuje w dwóch listach, a 6 i 7 — w żadnej,

`wIluListach 7 [[1,2,3],[3,4,5],[5,6,1],[1,7,4],[3,7,6],[2,7,5],[2,4,6]]` ma zwrócić `[3,3,3,3,3,3,3]`, bo każda z liczb od 1 do 7 występuje w dokładnie trzech listach.

Zadanie 3. *Wierzba rosochata* to struktura danych, która pozwala na: dołączenie gałęzi (tj. listy elementów), usunięcie ostatnio dołączonej gałęzi, dołączenie elementu do ostatnio dołączonej gałęzi, usunięcie ostatnio dołączonego elementu (pod warunkiem, że od ostatniego dołączenia elementu nie były wykonywane operacje na gałęziach), podanie liczby gałęzi oraz zamianę całej struktury na listę w taki sposób, że ostatnio dołączona gałąź pojawia się w liście przed pozostałymi gałęziami. Zamiana na listę powinna być wykonalna w czasie liniowym względem łącznej liczby elementów, natomiast pozostałe operacje — w czasie stałym. Zdefiniować typ Wr a, służący do przechowywania elementów typu a w wierzbie rosochatej, oraz następujące funkcje, realizujące opisane wyżej operacje z odpowiednią złożonością:

```
dg :: Wr a -> [a] -> Wr a
ug :: Wr a -> Wr a
de :: Wr a -> a -> Wr a
ue :: Wr a -> Wr a
lg :: Wr a -> Integer
wr2l :: Wr a -> [a]
```