Programowanie funkcyjne — kolokwium nr 1, 10.12.2015

Instrukcja: Rozwiązania zadań należy przesłać do godziny 13:15 na adres `kolokwium.pf@gmail.com` (decyduje data stempla googlowego). Każde zadanie należy przesłać w oddzielnym pliku: Zadanie1.hs, Zadanie2.hs i Zadanie3.hs (jeśli zadania nie udało się rozwiązać, należy przesłać pusty plik). Plików nie należy zipować. 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.
Uwaga: Korzystanie z internetu poza wyznaczonym czasem skutkuje automatycznym otrzymaniem 0 punktów.

Zadanie 1. Każde słowo złożone z liter a i b przekształcamy idąc od lewej według następujących reguł: $ab \to a, ba \to b, bb \to a, aa \to aaa$ (jeśli słowo ma nieparzystą długość, ostatnią literę przepisujemy). Napisać funkcję `dlugosc`, zwracającą liczbę iteracji powyższych reguł, które prowadzą do słowa złożonego z samych liter a lub słowa długości mniejszej niż 2. Przykładowo:

`dlugosc "abaaa" = 1`, bo $|ab|aa|a \to |a|aaa|a$
`dlugosc "abba" = 2`, bo $|ab|ba| \to |a|b|$ oraz $|ab| \to a$
`dlugosc "babba" = 3`, bo $|ba|bb|a \to |b|a|a$, $|ba|a \to |b|a$ oraz $|ba| \to b$

Zadanie 2. (a) Dla dwóch liczb całkowitych $a, b > 1$ oznaczamy przez $\mathrm{val}(a, b)$ największą potęgę liczby b, która dzieli a, np. $\mathrm{val}(56, 2) = 3$, bo 2^3 dzieli 56, ale 2^4 nie dzieli 56; podobnie $\mathrm{val}(56, 3) = 0$, bo 3^1 nie dzieli 56. Napisać funkcję:

```
val :: Integer -> Integer -> Integer
```

która dla podanych a i b oblicza $\mathrm{val}(a, b)$.
(b) Za pomocą powyższej funkcji `val` napisać funkcję:

```
g :: Integer -> Integer -> [Integer]
```

która dla podanych $k > 1$ oraz $v \geqslant 0$ zwraca listę (nieskończoną, w dowolnej kolejności) liczb naturalnych $n > 1$, takich że $\mathrm{val}(n, k) = v$. Przykładowo:

```
g 2 0 = [3,5,7,9,...]
g 3 1 = [3,6,12,15,21,...]
```

Zadanie 3. Kłosem nazywamy strukturę danych o funkcjonalności przypominającej listę, która umożliwia dokładanie elementów na początek i na koniec w czasie stałym, a ponadto odczytanie elementów po kolei. Stworzyć typ `Klos a` przechowujący elementy typu `a`. Zdefiniować funkcje:

```
wnpk :: Klos a -> a -> Klos a
wnkk :: Klos a -> a -> Klos a
k2list :: Klos a -> [a]
```

Funkcje mają, odpowiednio, wstawiać element na początek i na koniec kłosa oraz zamieniać kłos na listę. W ostatniej funkcji nie nakładamy ograniczenia na złożoność.