Haskell – wprowadzenie c.d.

zadania

Zadanie 1

Napisać funkcję o sygnaturze

```
jestPunktemStalym :: (Eq a) => a -> (a -> a) -> Bool
```

sprawdzającą, czy punkt podany jako pierwszy argument jest punktem stałym funkcji podanej jako drugi argument.

Zadanie 2

Napisać funkcję dwuargumentową (możliwie najogólniejszą), która przyjmuje liczby x, y oraz zwraca napis, który informuje nas, czy iloczyn x i y jest większy od ich sumy. W napisie mają pojawić się liczby x i y.

Zadanie 3

Zdefiniować nieskończoną listę dodatnich rozwiązań układu kongruencji:

$$\begin{cases} x \equiv 5 \pmod 6 \\ x \equiv 3 \pmod 7 \\ x \equiv 5 \pmod 8 \end{cases}$$

Zadanie 4

Zdefiniować funkcję `dzielniki :: Int -> [Int]` zwracającą listę dzielników liczby z wykorzystaniem funkcji z dzisiejszych materiałów (por. zadanie do pierwszych ćwiczeń z liczbą doskonałą).

Zadanie 5

Napisać funkcję o sygnaturze

```
ktoraCwiartka :: [(Int,Int)] -> Int
```

przyjmującą listę punktów na płaszczyźnie i zwracającą numer ćwiartki, w której jest najwięcej punktów (punkty na osiach liczymy dla obu przylegających ćwiartek, w szczególności (0,0) liczymy do każdej ćwiartki). Jeśli w kilku ćwiartkach występuje tyle samo punktów, zwrócić numer dowolnej z nich.

Zadanie 6

Napisać funkcję o sygnaturze

```
podlisty :: [Integer] -> [[Integer]]
```

która przyjmuje listę liczb całkowitych i zwraca listę wszystkich jej podlist. Wykorzystać ją do napisania funkcji o sygnaturze

```
podlistyDlugosci :: Integer -> [Integer] -> [[Integer]]
```

która przyjmuje liczbę k oraz listę liczb całkowitych i zwraca listę wszystkich podlist podanej listy o długości k.

Zadanie 7

Napisać funkcję

```
przeksztalcListe :: (Int -> Int) -> [Int] -> [Int]
```

przyjmującą funkcję oraz listę liczb naturalnych, która wykonuje podaną funkcję na każdym elemencie listy (czyli dla funkcji f oraz listy $[x_1, \ldots, x_n]$ mamy otrzymać $[f(x_1), \ldots, f(x_n)]$). Wykorzystać ją do podniesienia każdej z liczb na liście [1..10] do kwadratu.