Języki i paradygmaty programowania
Lista 3 (elementy języka Haskell)

Przemysław Kobylański

Zaprogramuj w języku Haskell rozwiązania poniższych zadań. Na ocenę dostateczną trzeba rozwiązać wszystkie zadania bez gwiazdek. Na ocenę dobrą trzeba dodatkowo rozwiązać wszystkie zadania z jedną gwiazdką. Na ocenę bardzo dobrą trzeba rozwiązać wszystkie zadania.

Zadanie 1

Stosując ogólną postać list (*list comprehension*) zdefiniuj funkcję *scalarproduct* :: $Num\ a \Rightarrow [a] \to [a] \to a$ obliczającą iloczyn skalarny dwóch wektorów reprezentowanych listami liczb.

Przykład

```
> scalarproduct [1,2] [3,4]
11
> scalarproduct [1,0] [0,1]
0
> scalarproduct [1,2,3] [3,2,1]
10
```

Wskazówka

Przydatne mogą okazać się funkcje *sum* i *zip*.

Zadanie 2*

Używając funkcji bibliotecznych, zdefiniuj funkcję $split : [a] \to ([a], [a])$, która rozrzuca elementy listy na dwie listy mniej więcej równej długości[1].

Przykład

```
> split []
([], [])
> split [1]
([1],[])
> split [1,2]
([1],[2])
> split [1,2,3]
([1,3],[2])
> split [1,2,3,4]
```

[1] Długości obu list w wyniku nie powinny różnić się o więcej niż 1.

```
([1,3],[2,4])
> split [1,2,3,4,5]
([1,3,5],[2,4])
> split [1,2,3,4,5,6]
([1,3,5],[2,4,6])
```

Zastanów się nad czasową złożonością obliczeniową Twojej funkcji. Czy dałoby się napisać ją lepiej?

Wskazówka

Przydatna może się okazać fraza **where**.

Zadanie 3**

Zdefiniuj leniwą funkcję *permutacje* :: $Eq\ a \Rightarrow [a] \rightarrow [[a]]$, której wartością jest lista złożona ze wszystkich permutacji jej argumentu.

Przykład

```
> let x = permutacje [1..25]
> take 5 x
[[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25],
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,25,24],
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,23,25],
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,23],
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,25,23,24]]
```

Wskazówa

Przydatna może okazać się postać ogólna listy (*list comprehension*) oraz funkcja *delete*, którą znajdziesz w module *Data.List*.

Literatura

[1] G. Hutton. Programming in Haskell. Cambridge University Press, 2007.