Kurs języka Haskell 2024/25

LISTA NR 1 (TERMIN: 12.10.2024, godz 5:00)

Uwaga: Wszystkie rozwiązania należy umieścić w jednym module o nazwie `Lista1` zachowując poniższe sygnatury.

Zadanie 1. Zdefiniuj funkcję

```
factorial :: Integer → Integer
```

obliczającą silnię argumentu. Na przykład:

```
ghci> factorial 12
479001600
```

Zadanie 2. Trójargumentowa funkcja $\varphi : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ wymyślona przez Wilhelma Ackermanna zdefiniowana jest jako $\varphi(m, n, p) =$

$$\begin{cases} m+n & \text{gdy } p = 0 \\ 0 & \text{gdy } n = 0 \text{ oraz } p = 1 \\ 1 & \text{gdy } n = 0 \text{ oraz } p = 2 \\ m & \text{gdy } n = 0 \text{ oraz } p > 2 \\ \varphi(m, \varphi(m, n-1, p), p-1) & \text{w p.p.} \end{cases}$$

Zdefiniuj funkcję

```
ack :: Integer → Integer → Integer → Integer
```

obliczającą wartość funkcji φ, np.

```
ghci> ack 2 2 3
16
ghci> ack 2 3 3
65536
```

Zadanie 3. Zdefiniuj funkcję

```
removeVowels :: String → String
```

która usuwa samogłoski z ciągu znaków. Na przykład:

```
ghci> removeVowels "Ala ma kota"
"l m kt"
```

Wskazówka: Pamiętaj, że stringi w Haskellu to listy znaków, przydać się więc mogą standardowe funkcje z biblioteki:

```
filter :: (a → Bool) → [a] → [a]
elem :: a → [a] → Bool
```

Zadanie 4. Zdefiniuj funkcje

```
everyOtherIn :: [a] → [a]
everyOtherEx :: [a] → [a]
merge :: [a] → [a] → [a]
```

takie, że:

- everyOtherIn pozostawia co drugi element na liście, począwszy od pierwszego,
- everyOtherEx pozostawia co drugi element na liście, począwszy od drugiego,
- merge tworzy listę naprzemiennie pobierając elementy z obu list, począwszy od listy będącej pierwszym argumentem. Jeśli jedna z list „skończy się" jako pierwsza, resztę wyniku stanowi druga lista.

Na przykład:

```
ghci> everyOtherIn "Ala ma kota"
"Aam oa"
ghci> everyOtherEx "Ala ma kota"
"l akt"
ghci> merge (everyOtherIn "Ala ma kota")
            (everyOtherEx "Ala ma kota")
"Ala ma kota"
ghci> merge "Haskell" "Ala ma kota"
"HAalsak emlal kota"
```

Zadanie 5. Zaimplementuj funkcję

```
transpose :: [[a]] → [[a]]
```

dokonującą transpozycji macierzy reprezentowanej jako lista list. Na przykład:

```
ghci> transpose ["rok","ala","but","las"]
["rabl","olua","kats"]
```

Zadanie 6. Skorzystaj z listy liczb pierwszych zdefiniowanej w kodzie opublikownym w SKOS-ie, żeby zaimplementować funkcję

```
primeFactors :: Integer → [Integer]
```

rozkładającą liczbę na czynniki pierwsze. Np.:

```
ghci> take 10 (map primeFactors [1..])
[[],[2],[3],[2,2],[5],[2,3],[7],[2,2,2],[3,3],[2,5]]
```