Programowanie funkcyjne i współbieżne
Lista 4

Na wykładzie zostały zdefiniowane drzewa binarne:

```
sealed trait BT[+A]
case object Empty extends BT[Nothing]
case class Node[+A](elem:A, left:BT[A], right:BT[A]) extends BT[A]

val t = Node(1, Node(2, Empty, Node(3, Empty, Empty)), Empty)
```

1. Napisz funkcję sumBT: BT[Int] => Int, która oblicza sumę liczb całkowitych, przechowywanych w węzłach drzewa, np. sumBT(t) == 6

2. Napisz funkcjonał

foldBT[A, B](f: A => (B, B) => B)(acc: B)(bt: BT[A]): B

uogólniający funkcję sumowania wartości z węzłów drzewa binarnego tak, jak funkcjonał foldRight uogólnia funkcję sumowania elementów listy. Typ (B, B) to typ pary akumulatorów dla lewego i prawego poddrzewa.

3. Wykorzystaj foldBT do zdefiniowania:
   a) sumy liczb całkowitych sumBTfold: BT[Int] => Int, np. sumBTfold(t) == 6
   b) listy wartości pamiętanych w węzłach drzewa w obejściu:
      infiksowym - inorderBTfold[A](bt: BT[A]): List[A]
      prefiksowym - preorderBTfold[A](bt: BT[A]): List[A]
      postfiksowym - postorderBTfold[A](bt: BT[A]): List[A]
      np. inorderBTfold(t) == List(2, 3, 1)
      preorderBTfold(t) == List(1, 2, 3)
      postorderBTfold(t) == List(3, 2, 1)

4. Wykorzystując foldBT zdefiniuj funkcjonał

mapBT[A, B](f: A => B)(bt: BT[A]): BT[B]

aplikujący daną funkcję do wartości we wszystkich węzłach drzewa.
np. mapBT[Int, Int](v => 2 * v)(t) == Node(2,Node(4,Empty,Node(6,Empty,Empty)),Empty)

5. Na wykładzie zostały zdefiniowane grafy:

```
sealed trait Graphs[A]
case class Graph[A](succ: A => List[A]) extends Graphs[A]
```

Napisz funkcję: pathExists[A](g: Graph[A])(from: A, to: A): Boolean
sprawdzającą, czy istnieje droga pomiędzy zadanymi wierzchołkami grafu.
np. dla poniższego grafu g:

```
pathExists(g)(4, 1)
!pathExists(g)(0, 4)
!pathExists(g)(3, 0)
pathExists(g)(2,2)
!pathExists(g)(0,0)
```

```
val g = Graph((i: Int) =>
  i match {
    case 0 => List(3)
    case 1 => List(0, 2, 4)
    case 2 => List(1)
    case 3 => List(5)
    case 4 => List(0, 2)
    case 5 => List(3)
    case n => throw new NoSuchElementException(s"Graph g: node $n doesn't exist")
})
```