Paradygmaty programowania - ćwiczenia
Lista 5

Na wykładzie zostały zdefiniowane listy leniwe.
OCaml: `type 'a llist = LNil | LCons of 'a * (unit -> 'a llist);;`

1. (OCaml)
   Zdefiniuj funkcję, która dla danej nieujemnej liczby całkowitej *k* i listy leniwej $[x_1, x_2, x_3, \ldots]$ zwraca listę leniwą, w której każdy element jest powtórzony *k* razy, np. dla *k*=3:
   $$[x_1, x_1, x_1, x_2, x_2, x_2, x_3, x_3, x_3, \ldots]$$
   Uwaga. Dla zwiększenia czytelności zastosowano tu notację dla zwykłych list.

2. Zdefiniuj leniwą listę liczb Fibonacciego *lfib* : `int llist` (OCaml) i *lfib* : `Stream[Int]` (Scala).

3. Polimorficzne leniwe drzewa binarne można zdefiniować następująco:
OCaml: `type 'a lBT = LEmpty | LNode of 'a * (unit ->'a lBT) * (unit -> 'a lBT);;`
Scala:

```
sealed trait lBT[+A]
case object LEmpty extends lBT[Nothing]
case class LNode[+A](elem:A, left:()=>lBT[A], right:()=>lBT[A]) extends lBT[A]
```

a) Napisz funkcję `lTree` , która dla zadanej liczby naturalnej *n* konstruuje nieskończone leniwe drzewo binarne z korzeniem o wartości *n* i z dwoma poddrzewami `lTree (2*n)` oraz `lTree( 2*n+1)`. To drzewo jest przydatne do testowania funkcji z następnego podpunktu.

b) Napisz funkcję, tworzącą leniwą listę w OCamlu (strumień w Scali), zawierającą wszystkie wartości węzłów leniwego drzewa binarnego.
Wskazówka: zastosuj obejście drzewa wszerz, reprezentując kolejkę jako zwykłą listę.