**Lista 5**

Zdefiniowane są następujące typy:

```
type 'a llist = LNil | LCons of 'a * (unit -> 'a llist);;
type 'a lBT = LEmpty | LNode of 'a * ('a lBT Lazy.t) * ('a lBT Lazy.t);;
```

**Uwaga.** Testy należy przeprowadzić dla list nieskończonych i skończonych.

1. Zdefiniuj nieskończony ciąg liczb Fibonacciego *lFib: int llist*.

2. Napisz funkcję *lrepeat : (int ->int) -> 'a llist -> 'a llist*, która dla danej funkcji *f:int -> int* i listy leniwej $[x_0, x_1, x_2, x_3, \ldots]$ : *'a llist* zwraca listę leniwą , w której każdy element $x_i$ jest powtórzony *f(i)>0* razy, np.
   *lrepeat* (fun i -> i+1) $[x_0, x_1, x_2, x_3, \ldots]$ => $[x_0, x_1, x_1, x_2, x_2, x_2, x_3, x_3, x_3, x_3, x_4, x_4, \ldots]$
   *Uwaga*. Dla lepszej czytelności zastosowano tu notację dla zwykłych list.

3. Zdefiniuj funkcję *sublist: xs:int list -> ll:'a llist -> 'a llist*, która dla listy *xs: int list* oraz dla listy leniwej *ll*=$[x_0, x_1, x_2, \ldots]$ zwraca listę leniwą, z której usunięto wszystkie elementy $x_i$, których indeks *i* znajduje się na liście *xs*.
   Przykład: *xs* = [1; 4; 7; 2] *ll* = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, ...]
   Wynik: [10, 13, 15, 16, 18, 19, 20, 21, ...]

4. Zdefiniuj funkcję *toLBST : 'a list -> 'a lBT*, która z listy tworzy leniwe binarne drzewo poszukiwań. W teście obejdź to drzewo infiksowo i zwróć listę wartości, przechowywanych w węzłach drzewa.