

## Zadanie 1.

W pliku `bsttree.hs` uzupełnić kod funkcji `insertElement` tak, aby wstawiała ona element do drzewa BST. Następnie napisać funkcję `makeTree`, która tworzy drzewo BST z elementów podanych na liście.

**Uwaga.** Można założyć, że wstawiane będą parami różne elementy.

## Zadanie 2.

W pliku `expr.hs` zdefiniowany jest typ danych odpowiadający wyrażeniom, w których możliwymi do wykonania działaniami są dodawanie/odejmowanie oraz mnożenie/dzielenie. Napisać funkcję `eval`, która wylicza wartość podanego wyrażenia oraz podać jej najbardziej ogólną sygnaturę.

## Zadanie 3.

Uzasadnić (np. rysując drzewo), że

```
foldl (\x y -> x-2*y) 10 [10,20,1,5] = -62
```

```
foldr (\x y -> x-2*y) 10 [10,20,1,5] = 94
```

## Zadanie 4.

Wielomian  $f(x) = a_d x^d + a_{d-1} x^{d-1} + \dots + a_0$  reprezentujemy za pomocą listy  $[a_d, a_{d-1}, \dots, a_0]$ . Przy pomocy funkcji `foldl/r` napisać funkcję `horner f x0`, która oblicza  $f(x_0)$  za pomocą schematu Hornera.


## Zadanie 5.

Napisać funkcję `freq :: [a] -> [(a, Int)]`, która zwraca informacje o liczbie wystąpień każdego elementu z listy podanej jako argument. Na przykład

```
freq "alamakota" =  
[( 'a', 4), ( 'l', 1), ( 'm', 1), ( 'k', 1), ( 'o', 1), ( 't', 1)].
```

W rozwiązaniu należy użyć jednej z funkcji `foldl/foldr`.<sup>1</sup>

---

<sup>1</sup>Źródło tego problemu zostanie podane w terminie późniejszym. 

## Zadanie 6.

Napisać funkcje

- ▶ `insertElement :: Ord a => a -> [a] -> [a]`, który wstawia element na "właściwe" miejsce do listy. "Właściwe" = takie, że utworzona lista reprezentuje ciąg posortowany (np. rosnąco).
- ▶ `insertionSort :: Ord a => [a] -> [a]`, która sortuje podaną listę przez wstawianie.

W rozwiązaniu należy (przynajmniej raz) w istotny sposób użyć funkcji `foldl/r`.

## Zadanie domowe

Zdefiniować typ `Complex`, który będzie przechowywał liczby zespolone. Uczynić go instancją klas `Num` oraz `Show` tak, aby można było prowadzić operacje na liczbach zespolonych oraz wyświetlać je w postaci  $a+b*I$ .

**Uwaga.** Funkcja `signum` może zostać zaimplementowana *dowolnie*. Natomiast przy implementacji funkcji `fromInteger` pomocna może być funkcja `fromIntegral` (zob. np. Hoogle).

## Zadanie domowe

W zadaniu tym rozważamy drzewa BST, o których wiemy, iż nie zawierają powtarzających się elementów. Napisać funkcję `subSize :: Ord a => a -> BST a -> Int`, która dla wywołania `subTree x t` zwraca liczbę wierzchołków poddrzewa drzewa `t` o korzeniu w `x`.

Na przykład dla poniższego drzewa `subTree 30 t = 4` (na zielono zaznaczono odpowiednie poddrzewo).

