Изпит по Функционално програмиране
Специалност „Информационни системи“, I курс, 05.07.2022 г.

Задача 1. Чрез използването на `n` на брой кубове е построена сграда. Кубът, намиращ се най-отдолу, т.е. основата, е с обем n^3. Кубът, който е върху него, е с обем $(n-1)^3$. Кубът, поставен най-отгоре, има обем 1^3.
Обемът на цялата сграда е: $n^3 + (n-1)^3 + ... + 1^3 = m$.
Да се дефинира функция **`findNb :: Integer -> Integer`**, която по дадено `m` да връща броя кубове `n`, необходими за построяване на сградата. Ако такова `n` не съществува, да се връща `-1`.

Примери:

```
findNb 1071225            → 45
findNb 40539911473216     → 3568
findNb 135440716410000    → 4824
findNb 4183059834009      → 2022
findNb 91716553919377     → -1
findNb 24723578342962     → -1
```

Задача 2. Да се дефинира функция **`prodEvens :: (Num a) => [a] -> a`**, която приема списък от числа и намира произведението на числата, намиращи се на позиции с четен индекс в списъка. Списъкът е индексиран от 0. Задачата да се реши чрез **foldr**!

Примери:

```
prodEvens [1,2,3,4,5,6]   → 15
prodEvens [7.66,7,7.99,7] → 61.2034
```

Задача 3. Да се дефинира функция **`bounds :: [Int] -> (Int, Int)`**, която приема несортиран списък от цели числа и връща двойка, съдържаща индексите на елементите, формиращи най-късата последователност от елементи, която трябва да бъде сортирана, за да се сортира целият списък. Може да се счита, че подаденият списък не е сортиран.

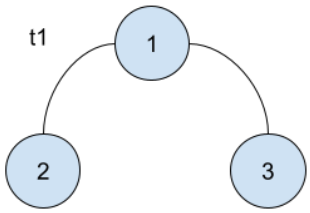
Примери:

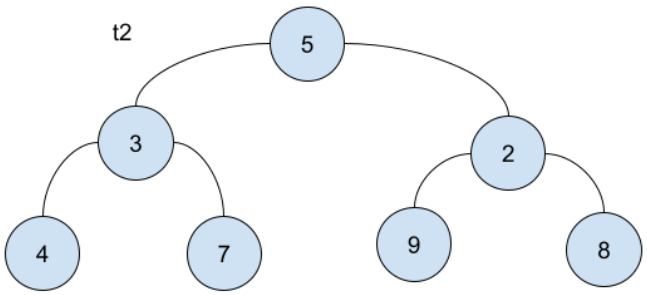
```
bounds [3, 7, 5, 6, 9]  → (1, 3)
bounds [3, 4, 5, 6, -1] → (0, 4)
bounds [3, -4, 5, 6]    → (0, 1)
```

Задача 4. Дефиниран е полиморфен алгебричен тип **`BTree a`**, описващ двоично дърво: **`data BTree a = Nil | Node a (BTree a) (BTree a)`**.

Да се дефинира функция **`areCousins :: (Eq a) => BTree a -> a -> a -> Bool`**, която по подадено двоично дърво, съдържащо възли с уникални стойности, и две стойности от типа на тези във възлите на дървото проверява дали възлите с подадените стойности са братовчеди в подаденото дърво. Два възела в едно дърво са братовчеди, ако са разположени на една и съща дълбочина и имат различни родители.

Примери:






```
t1 :: BTree Int
t1 = Node 1 (Node 2 Nil Nil)
            (Node 3 Nil Nil)

t2 :: BTree Int
t2 = Node 5 (Node 3 (Node 4 Nil Nil)
                    (Node 7 Nil Nil))
            (Node 2 (Node 9 Nil Nil)
                    (Node 8 Nil Nil))

areCousins t1 2 3 → False
areCousins t2 8 4 → True
```