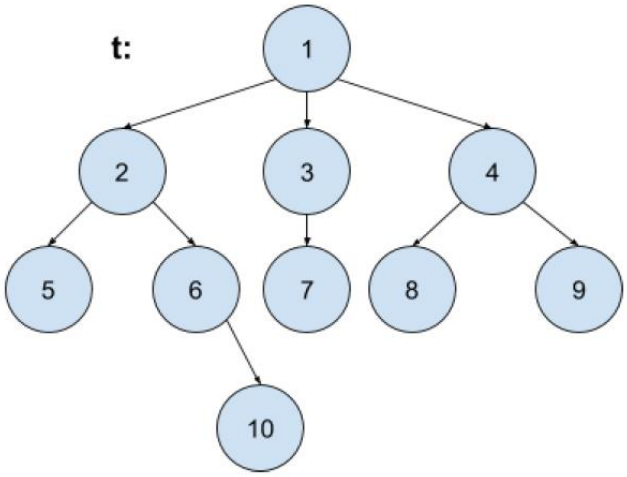
КОНТРОЛНА РАБОТА № 2 ПО ФУНКЦИОНАЛНО ПРОГРАМИРАНЕ
Специалност „Информационни системи“, 1-ви курс
02.06.2019 г.

Задача 1. Дефинирайте функция **specialSum :: (Int -> Int) -> [Int] -> ((Int -> Bool) -> Int)**, която получава като аргументи едноместна целочислена функция **f** и списък от естествени числа **xs**. Функцията трябва да върне нова функция с един аргумент - функция предикат **p**. Върнатата функция трябва да намира сумата на квадратите на тези числа **x** от списъка **xs**, за които композицията **p (f x)** е истина.
Примери:

```
((specialSum (5 -) [1..10]) (> 0)) → 30

((specialSum (\x -> x + 1) [(-5)..5]) odd) → 40
```

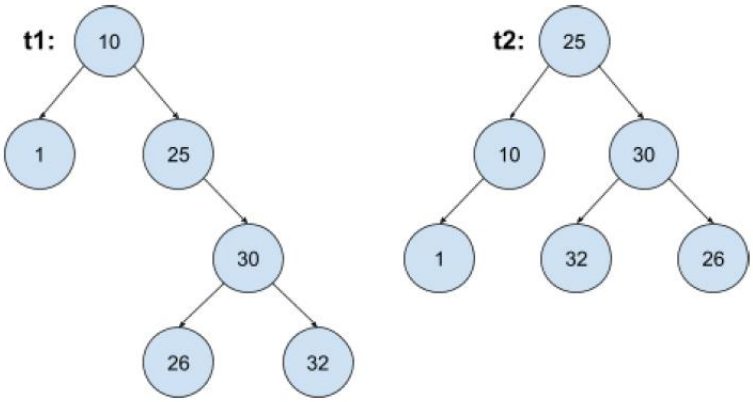
Задача 2. Дефинирайте функция **findUncles tree node**, която за дадени дърво от естествени числа **tree** и връх **node** на **tree** намира списък от всички чичовци на **node** в **tree**. Дървото **tree** е представено чрез асоциативен списък, ключове в който са върховете на дървото, а асоциираната с даден ключ стойност е списък от синовете на съответния връх.
Примери:



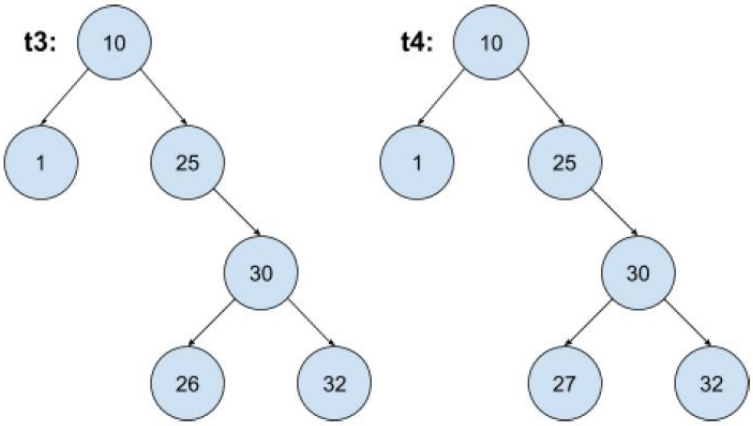

```
t :: [(Int, [Int])]
t = [(1,[2,3,4]),(2,[5,6]),(3,[7]),(4,[8,9]),
     (5,[]),(6,[10]),(7,[]),(8,[]),(9,[]),(10,[])]

(findUncles t 5) → [3,4]

(findUncles t 7) → [2,4]

(findUncles t 10) → [5]
```

Задача 3. Нека двоично дърво да се представя като: **data** BTree = Nil | Node Int BTree BTree. Дефинирайте функция предикат **leavesAreEqual :: BTree -> BTree -> Bool**, която получава две двоични дървета **bt1** и **bt2** и проверява дали възходящата подредба на стойностите в листата на **bt1** е същата като възходящата подредба на стойностите в листата на **bt2**.

Примери:



(leavesAreEqual t1 t2) → True



(leavesAreEqual t3 t4) → False