Задание № 2

1. Написать функцию prodIndices :: [Integer] -> [Int], которая по списку целых чисел выдает список индексов тех его элементов, которые являются произведениями двух соседних элементов (если один из соседних элементов отсутствует, то он полагается равным единице). Например, prodIndices [1, -2, 6, -3, 0, 0] должен выдать список [2, 4, 5], поскольку элементы с индексами 2, 4 и 5 (и только они) являются произведениями соседних элементов: 6 = (-2)\*(-3), 0 = (-3)\*0, 0 = 0\*1. В этом задании требуется по существу дела использовать функции высших порядков. Решения, основанные на чистой рекурсии без функций высших порядков будут оцениваться ниже.
2. Определите тип данных для представления двоичного дерева поиска

data Tree key value = ..

для которого требуется реализовать стандартный набор методов для манипуляций с таким деревом:

get :: Ord k => k -> Tree k v -> Maybe v

put :: Ord k => (k, v) -> Tree k v -> Tree k v

remove :: Ord k => k -> Tree k v -> Tree k v

keys :: Tree k v -> [k]

values :: Tree k v -> [v]

Все вышеперечисленные функции имеют обычный для операций над отображениями смысл: найти значение по ключу (get), добавить или заменить существующую пару <ключ,значение> (put), удалить пару с заданным ключом (remove), выдать множество (список) всех ключей (keys) и выдать список всех значений (values).

Еще одна функция, которую требуется реализовать, это

pairsBy :: (k -> Bool) -> Tree k v -> [(k, v)],

которая выдает множество тех пар отображения, ключи которых удовлетворяют критерию, заданному первым аргументом функции. Например, вызов pairsBy (<0) t  выдаст те пары отображения m, ключи которых меньше нуля