Функциональное программирование **ПМИ**

Домашнее задание 4.

Напомним, что типы `A` и `B` *изоморфны*, если существуют функции `f :: A -> B` и `g :: B -> A`, такие что верны равенства `f . g = id` и `g . f = id`.

1. Докажите, что для любых типов `A, B` изоморфны типы:

a) `Ordering` и `Either () Bool`;

b) `A` и `(A, ())`;

c) `(Maybe A, Maybe B)` и `Maybe (Either (A, B) (Either A B))`.

2. Определение натуральных чисел `data Nat = Z | S Nat` весьма неэффективно, поскольку представление числа n имеет размер $O(n)$. Предположим, определен тип:

```
data NatB = ZB | Db NatB | DbI NatB
```

Проинтерпретируйте значения этого типа как натуральные числа таким образом, чтобы запись числа n была размера $O(\log n)$ (тут достаточно неформального объяснения). Реализуйте функции `natb2nat :: NatB -> Nat` и `nat2natb :: Nat -> NatB` в соответствии с вашей интерпретацией и таким образом, что `natb2nat . nat2natb = id`. Получился ли у вас изоморфизм типов `Nat` и `NatB`?

3. Назовем *отрезком* списка его кусок от i-го до $(i + k)$-го элемента включительно. Реализуйте функцию `segs :: [a] -> [[a]]`, выводящую все отрезки данного списка в любом порядке. Например, список `segs "hello"` должен с точностью до перестановки совпадать со списком `["h", "e", "l", "l", "o", "he", "el", "ll", "lo", "hel", "ell", "llo", "hell", "ello", "hello"]`. Если все элементы входного списка были различны, повторов в полученном списке быть не должно. Если входной список бесконечен, допускается любое поведение функции.

4. Быть может, используя `zip`, реализуйте функцию `nrem :: Int -> [a] -> [a]`, удаляющую каждый n-ый (считая с первого), элемент списка, так что `nrem 3 [1,2,3,4,5,6,7] == [1,2,4,5,7]`. Если входной список бесконечен, допускается любое поведение функции.

5. Реализуйте функцию, устраняющую в списке все повторы (необязательно идущие подряд). Какая из повторяющихся копий оставляется, решите сами. Если входной список бесконечен, допускается любое поведение функции.

6. Реализуйте функцию `part :: Int -> Int -> [[Int]]`, т. ч. `part m n` есть список всех разбиений `[x1,...,xm]` числа $n \geqslant 0$ в сумму $m > 0$ целых неотрицательных слагаемых. Порядок слагаемых в разбиении важен, а порядок разбиений в выводе функции — нет.

7. Используя `foldl` и `foldr`, реализуйте библиотечные функции:

a) `map`;

b) `filter`;

c) `all`; (*постарайтесь не использовать ни одной локальной переменной типа элемента списка, а применять оператор композиции*)

d) `any`. (*то же*)

8. Реализуйте библиотечные функции (прежде опишите словесно, что они делают):

a) `takeWhile`;

b) `dropWhile`.

9. Библиотечная функция `foldr1` отличается от `foldr` тем, что не использует начального значения. Опишите словесно, как она работает, и реализуйте ее с помощью `foldr`. Используя `foldr1`, реализуйте функцию `lmax :: Ord a => [a] -> a`, которая ищет максимальный элемент в непустом списке.

10. Реализуйте функцию, возвращающую список всех префиксов строки–аргумента. Если входная строка бесконечна, допускается любое поведение функции.

11. Реализуйте функцию `rotts :: [a] -> [[a]]`, возвращающую всевозможные циклические сдвиги (перестановки) данного списка, с помощью:

a) `iterate`;

b) `scanl`.

В частности, должно быть `rotts [1,2,3] = [[1,2,3], [2,3,1], [3,1,2]]` с точностью до перестановки «внешнего» списка. Если входной список бесконечен, допускается любое поведение функции.

12. Реализуйте библиотечную функцию `unzip :: [(a,b)] -> ([a],[b])`, прежде описав словесно, что она делает, используя:

a) `map`;

b) `foldr`.