Функциональное программирование.

Домашнее задание 3.

1. Определите какие-нибудь (разумные) выражения, имеющие тип:

a) `(b -> c, a -> b) -> a -> c` (буквы a, b, c означают переменные по типам, по которым подразумеваются кванторы всеобщности);

b) `[Double -> Double] -> Int -> Int`.

2. Какой тип имеет терм `\f -> f $ ($) $ (.)`? (Желательно указать наиболее общий тип.) Объясните, как прийти к такому выводу без помощи компилятора.

3. Определите следующие булевы функции с помощью механизма сравнения с образцом (pattern matching), не используя какие-либо иные, уже определенные, функции:

a) исключающее или — используя не более трех образцов;

b) функция большинства maj_3 (возвращает значение большинства своих аргументов) — используя не более четырех обрацов.

4. Определите функцию `f :: Integer -> Integer`, такую что $f(n) = 0^{fib(n)} + 1^{fib(n)} + \ldots + n^{fib(n)}$ при всех $n \geq 0$, где $fib(n)$ есть n-е число Фибоначчи.

5. Определите бесконечный список всех пифагоровых троек, т. е. троек вида (x, y, z), где $x^2 + y^2 = z^2$, типа `(Integer,Integer,Integer)`.

6. Натуральные числа n и m *дружественные*, если сумма собственных (т.е. меньших n) делителей числа n равна m, и наоборот (например, 220 и 284 дружественные). Определите предикат, проверяющий пару натуральных чисел на дружественность.

7. Выясните, что делают библиотечные функции `curry` и `uncurry`, и реализуйте их.

8. Допустим, есть некоторые типы `A, B, C, D` и функции `g :: A -> B -> D` и `h :: D -> C`, а для функции `f :: A -> B -> C` выполнено тождество `f x y = h (g x y)`. Определите `f`, не упоминая локальных переменных (т. е. `x` и `y`), с помощью библиотечных функций `curry`, `uncurry` и `(.)`.

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`.

9. Докажите, что для любых типов `A, B, C` изоморфны типы

a) `C -> (A, B)` и `(C -> A, C -> B)`;

b) `C -> (B -> A)` и `(B,C) -> A`.

10. С помощью `type` определите тип `Bfn` булевых функций трех аргументов.

a) Определите на таких *функциях* структуру кольца, т. е. операции сложения и умножения *функций*, получающиеся поточечным применением исключающего или и конъюнкции соответственно, а также предикат равенства. Определите нуль, единицу и взятие противоположного элемента в таком кольце. Определите функцию, вычисляющую разумное представление элемента `Bfn` в виде строки.

b) Поместив в начале модуля следующие директивы компилятора:

```
{-# LANGUAGE TypeSynonymInstances #-}
{-# LANGUAGE FlexibleInstances #-}
```

(позволяющие объявить экземпляром (instance) класса тип-синоним, вроде `Bfn`), с помощью механизма `instance` сделайте `Bfn` экземпляром классов `Eq`, `Num` и `Show` (в последнем случае нужно определить метод `show :: Bfn -> String`). Определения соответствующих методов должны быть возможно более разумными.