Функциональное программирование (осенний семестр 2021 г.)

Письменный экзамен

Deadline: 28 декабря 2021 г., 20:00. Решения присылать по адресу `sk@mi-ras.ru`

1. Перепишите следующий код через оператор связывания `>>=`:

```
do
    x <- [1,2,3]
    y <- [x,4]
    return (x,y)
```

   Чему равно значение этого выражения?

2. Рассмотрим следующий код:

```
import Control.Parallel
import Control.Monad.Random

faircoin = fromList [(True,0.5), (False,0.5)]
coins = replicateM 1000000 faircoin

gen1 = mkStdGen 42
gen2 = mkStdGen 23
array1 = evalRand coins gen1
array2 = evalRand coins gen2

op :: Bool -> Bool -> Bool
op = (||)

orFoldPar arr1 arr2 = b `par` (a `op` b) where
    a = foldl op False arr1
    b = foldl op False arr2

main = putStrLn $ show $ orFoldPar array1 array2
```

   Будет ли исполнение в два потока (`-N2`) быстрее, чем в один (`-N1`)? Объясните, почему. Если ответ «нет», поменяйте в коде не более двух символов так, чтобы исполнение в два потока стало быстрее.

3. Рассмотрим следующий код:

```
import Control.Monad.Random
dice = fromList [(1,1/6),(2,1/6),(3,1/6),(4,1/6),(5,1/6),(6,1/6)]
ncoin k = fromList [("heads",1/k),("tails",1-1/k)]
dcoin = dice >>= \x -> ncoin x
main = getStdGen >>= \gen -> putStrLn $ evalRand dcoin gen
```

С какой вероятностью исполнение этой программы выдаст “heads” ? (Ответ дайте точно, в виде обыкновенной дроби. Предполагаем, что `getStdGen` даёт «честный» генератор случайности.)

4. Пусть `m` — монада, т.е. даны `return :: a -> m a` и `(>>=) :: m a -> (a -> m b) -> m b`. Выразите через них `join :: m (m a) -> m a` и `fmap :: (a -> b) -> (m a -> m b)`.

5. Определим следующую операцию на типах: `m a = (a -> d) -> d`, где `d` — фиксированный тип. Определите `return` и `>>=`, чтобы `m` стала монадой. (Возможно, удобнее будет сначала определить `join`, а уже потом с его помощью `>>=`.)