Декларативное программирование

Списки

Сопоставление с образцом

Создание:

- Создать пустой список

```
[]
```

- Добавить в голову списка элемент

```
x : xs
```

Образцы:

```
firstElement :: [a] -> a
firstElement [] = error "Empty list!"
firstElement (x:xs) = x
```

Упражнение

На каких из следующих образцов удачно завершится сопоставление строки "Wow"

- (x : y)
- ((:) x y)
- [x, y]
- (x : y : z)
- [x, y, z]
- (x : [y, z])
- (x : y : z : [])
- (x : y : z : w)
- (x : y : z : w : [])

Какие значения будут связаны с переменными при удачном сопоставлении?

Упражнение

Напишите функцию, которая возвращает второй элемент списка:

```
secondElement :: [a] -> a
```

Упражнение

реализуйте следующие функции с использованием рекурсии:

```
sum' :: [Int] -> Int
replicate' :: Int -> a -> [a]
length' :: [a] -> Int
```

Упражнение

Какие выражения эквивалентны списку `[4,0,2,0]`?

1. `(((4 ++ 0) ++ 2) ++ 0)`
2. `(4 ++ (0 ++ (2 ++ 0)))`
3. `(4:(0:(2:(0:[]))))`
4. `(4:0:[]) ++ 2:(0:[])`
5. `((((4):0):2):0)`

Упражнение

Напишите функцию, которая возвращает последний элемент списка. Если список пустой функция должна вернуть значение по умолчанию

```
last' :: a -> [a] -> a
last' def xs = undefined
```

Например:

```
ghci> last’ 0 []
0
```

Упражнение

Напишите функцию map, которая применяет функцию к каждому элементу списка и помещает результат в итоговый список.

```
map' :: (a -> b) -> [a] -> [b]
map' f xs = undefined
```

Например:

```
ghci> map’ (^2) [1,2,3]
[1, 4, 9]
```

Упражнение

Напишите функцию map2, которая применяет функцию от двух аргументов к соответствующим значениям двух списков и помещает результат в итоговый список. Если списки имеют разные длины, «лишние» элементы игнорируются.

```
map2 :: (a -> b -> c) -> [a] -> [b] -> [c]
map2 f xs ys = undefined
```

Например:

```
ghci> map2 (+) [1,2,3] [4,5]
[5, 7]
```

Упражнение*

- Напишите функцию `groupElems` которая группирует подряд идущие одинаковые элементы в списке и возвращает список таких групп.

```
GHCi> groupElems []
[]
GHCi> groupElems [1,2]
[[1],[2]]
GHCi> groupElems [1,2,2,2,4]
[[1],[2,2,2],[4]]
GHCi> groupElems [1,2,3,2,4]
[[1],[2],[3],[2],[4]]
```

Упражнение*

Пусть есть список положительных достоинств монет coins, отсортированный по возрастанию. Воспользовавшись механизмом list comprehension, напишите функцию change, которая разбивает переданную ей положительную сумму денег на монеты достоинств из списка coins всеми возможными способами. Например, если coins = [2, 3, 7]:

GHCi> change 7

[[2,2,3],[2,3,2],[3,2,2],[7]]

Примечание. Порядок монет в каждом разбиении имеет значение, то есть наборы [2,2,3] и [2,3,2] — различаются.

change :: Integer -> [[Integer]]