Учреждение образования

«Полоцкий государственный университет имени Евфросинии Полоцкой»

Кафедра технологий программирования

Отчёт по курсу «[ФП (Haskell)\_23-ИТ-](https://stepik.org/class/17171)1» на stepik.org

Дисциплина «Функциональное программирование»

Выполнил: гр. 23-ИТ-1

Страпко В. Ю.

Проверил: старший преподаватель

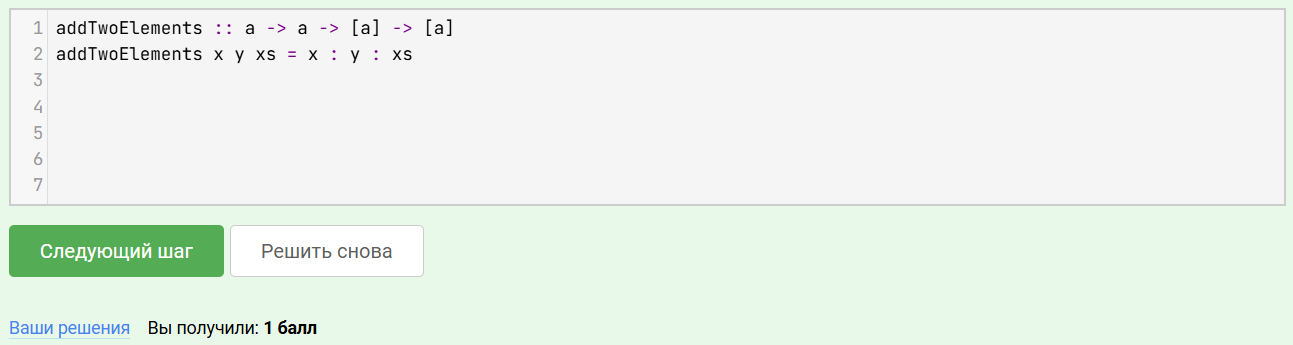
Забелендик О.Н.

Полоцк, 2025

Реализуйте функцию addTwoElements, которая бы добавляла два переданных ей значения в голову переданного списка.

GHCi> addTwoElements 2 12 [85,0,6]

[2,12,85,0,6]



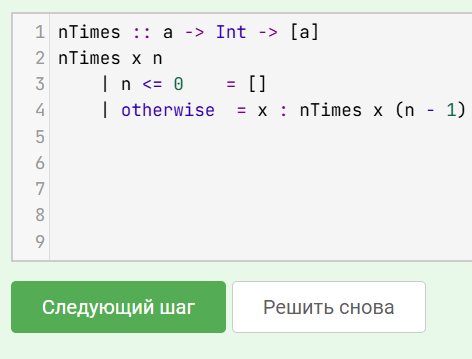
Реализуйте функцию nTimes, которая возвращает список, состоящий из повторяющихся значений ее первого аргумента. Количество повторов определяется значением второго аргумента этой функции.

GHCi> nTimes 42 3

[42,42,42]

GHCi> nTimes 'z' 5

"zzzzz"



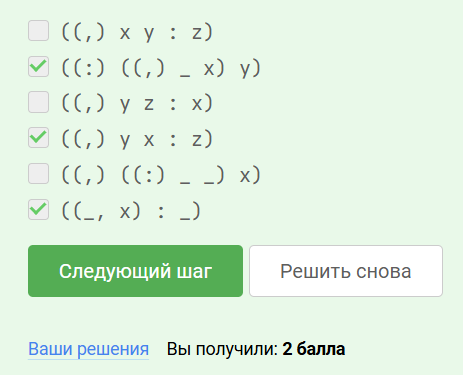
Исследуйте тип функции

sndHead = snd . head

и разберитесь, каково ее поведение. Эту функцию можно реализовать, используя сопоставление с образцом

sndHead некоторый\_образец = x

Отметьте те образцы, которые подходят для этой цели.

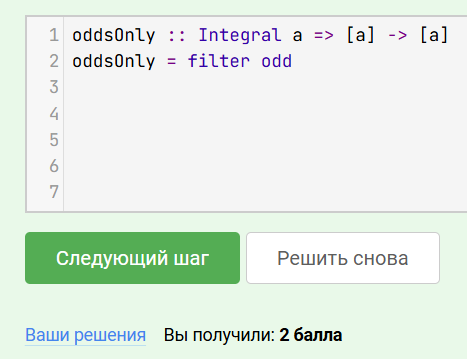


Сформируйте список целых чисел, содержащий только те элементы исходного списка, значение которых нечетно.

GHCi> oddsOnly [2,5,7,10,11,12]

[5,7,11]

Для анализа четности можно использовать функции odd и even стандартной библиотеки.



Реализуйте функцию isPalindrome, которая определяет, является ли переданный ей список палиндромом.

GHCi> isPalindrome "saippuakivikauppias"

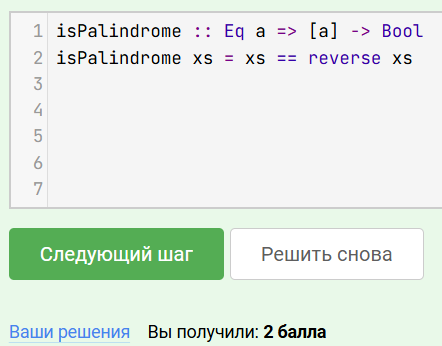
True

GHCi> isPalindrome [1]

True

GHCi> isPalindrome [1, 2]

False



Составьте список сумм соответствующих элементов трех заданных списков. Длина результирующего списка должна быть равна длине самого длинного из заданных списков, при этом «закончившиеся» списки не должны давать вклада в суммы.

GHCi> sum3 [1,2,3] [4,5] [6]

[11,7,3]



Напишите функцию groupElems которая группирует одинаковые элементы в списке (если они идут подряд) и возвращает список таких групп.

GHCi> groupElems []

[]

GHCi> groupElems [1,2]

[[1],[2]]

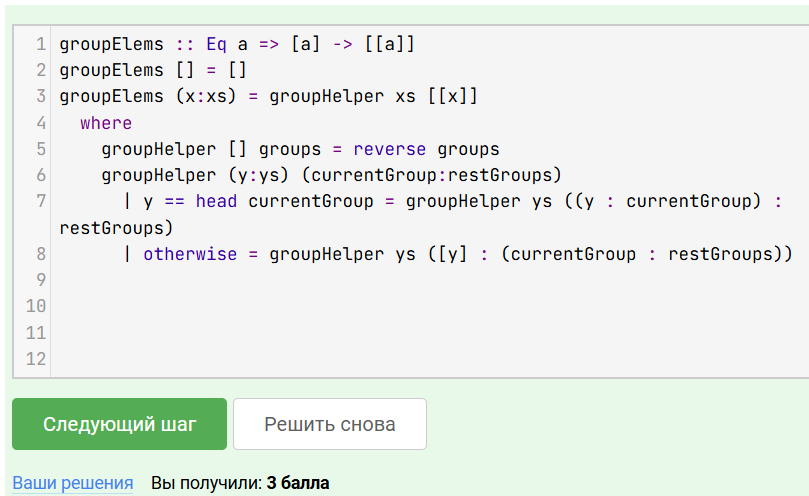
GHCi> groupElems [1,2,2,2,4]

[[1],[2,2,2],[4]]

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

[[1],[2],[3],[2],[4]]

Разрешается использовать только функции, доступные из библиотеки Prelude.



Напишите функцию readDigits, принимающую строку и возвращающую пару строк.

Первый элемент пары содержит цифровой префикс исходной строки, а второй - ее оставшуюся часть.

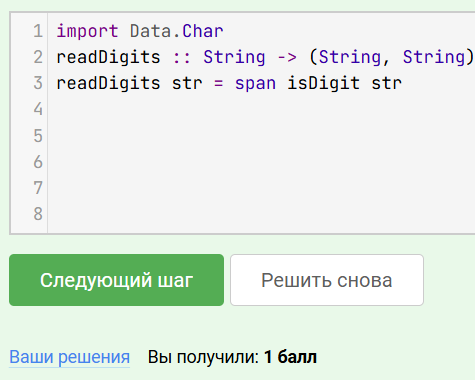
GHCi> readDigits "365ads"

("365","ads")

GHCi> readDigits "365"

("365","")

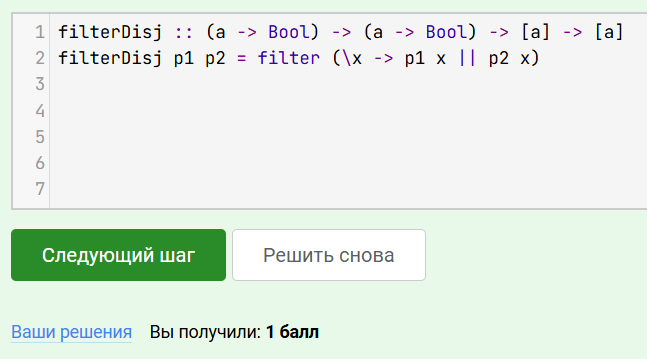
В решении вам поможет функция isDigit из модуля Data.Char.



Реализуйте функцию filterDisj, принимающую два унарных предиката и список, и возвращающую список элементов, удовлетворяющих хотя бы одному из предикатов.

GHCi> filterDisj (< 10) odd [7,8,10,11,12]

[7,8,11]

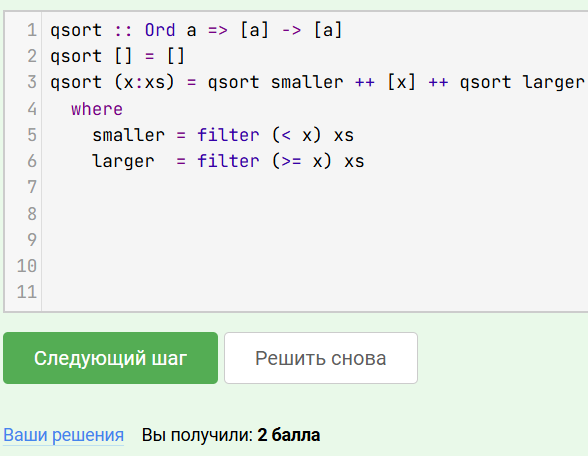


Напишите реализацию функции qsort. Функция qsort должная принимать на вход список элементов и сортировать его в порядке возрастания с помощью сортировки Хоара: для какого-то элемента x изначального списка (обычно выбирают первый) делить список на элементы меньше и не меньше x, и потом запускаться рекурсивно на обеих частях.

GHCi> qsort [1,3,2,5]

[1,2,3,5]

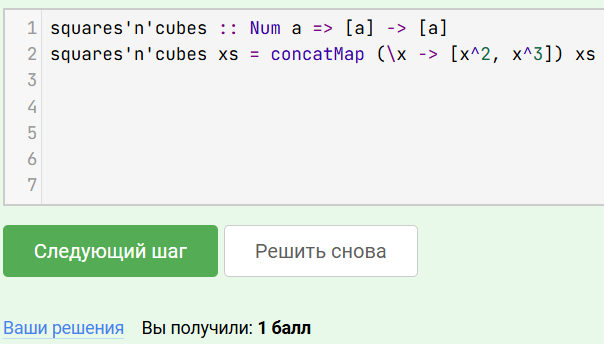
Разрешается использовать только функции, доступные из библиотеки Prelude.



Напишите функцию squares'n'cubes, принимающую список чисел, и возвращающую список квадратов и кубов элементов исходного списка.

GHCi> squares'n'cubes [3,4,5]

[9,27,16,64,25,125]

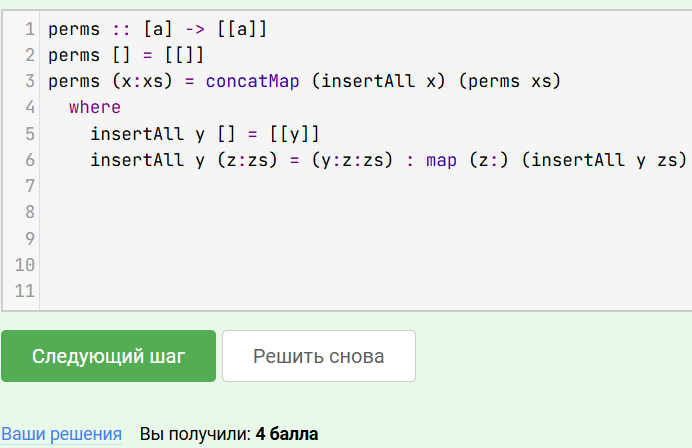


Воспользовавшись функциями map и concatMap, определите функцию perms, которая возвращает все перестановки, которые можно получить из данного списка, в любом порядке.

GHCi> perms [1,2,3]

[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

Считайте, что все элементы в списке уникальны, и что для пустого списка имеется одна перестановка.

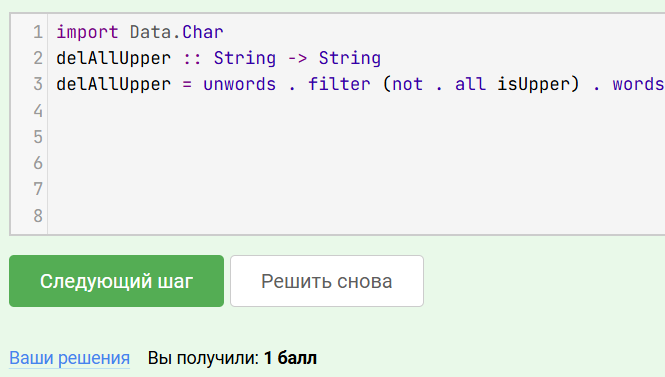


Реализуйте функцию delAllUpper, удаляющую из текста все слова, целиком состоящие из символов в верхнем регистре. Предполагается, что текст состоит только из символов алфавита и пробелов, знаки пунктуации, цифры и т.п. отсутствуют.

GHCi> delAllUpper "Abc IS not ABC"

"Abc not"

Постарайтесь реализовать эту функцию как цепочку композиций, аналогично revWords из предыдущего видео.



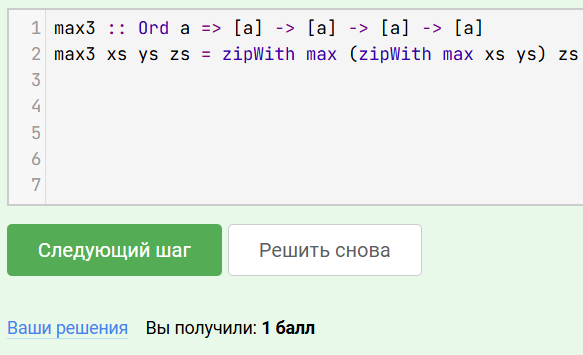
Напишите функцию max3, которой передаются три списка одинаковой длины и которая возвращает список той же длины, содержащий на k-ой позиции наибольшее значение из величин на этой позиции в списках-аргументах.

GHCi> max3 [7,2,9] [3,6,8] [1,8,10]

[7,8,10]

GHCi> max3 "AXZ" "YDW" "MLK"

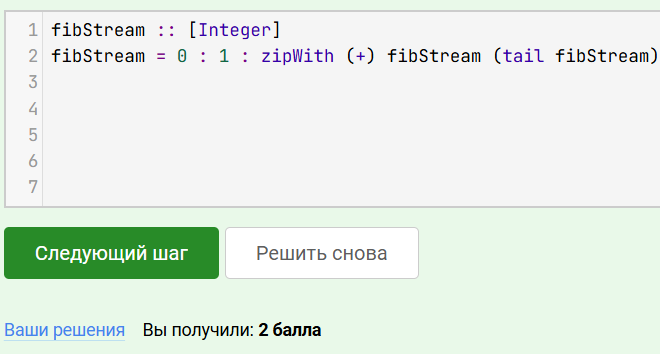
"YXZ"



Реализуйте c использованием функции zipWith функцию fibStream, возвращающую бесконечный список чисел Фибоначчи.

GHCi> take 10 $ fibStream

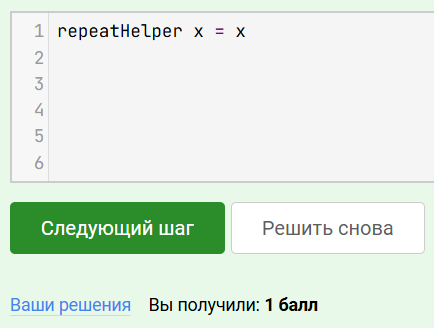
[0,1,1,2,3,5,8,13,21,34]



Предположим, что функция repeat, была бы определена следующим образом:

repeat = iterate repeatHelper

определите, как должна выглядеть функция repeatHelper.



Пусть задан тип Odd нечетных чисел следующим образом:

data Odd = Odd Integer

deriving (Eq, Show)

Сделайте этот тип представителем класса типов Enum.

GHCi> succ $ Odd (-100000000000003)

Odd (-100000000000001)

Конструкции с четным аргументом, типа Odd 2, считаются недопустимыми и не тестируются.

Примечание. Мы еще не знакомились с объявлениями пользовательских типов данных, однако, скорее всего, приведенное объявление не вызовет сложностей. Здесь объявляется тип данных Odd с конструктором Odd. Фактически это простая упаковка для типа Integer. Часть deriving (Eq, Show) указывает компилятору, чтобы он автоматически сгенерировал представителей соответствующих классов типов для нашего типа (такая возможность имеется для ряда стандартных классов типов). Значения типа Odd можно конструировать следующим образом:

GHCi> let x = Odd 33

GHCi> x

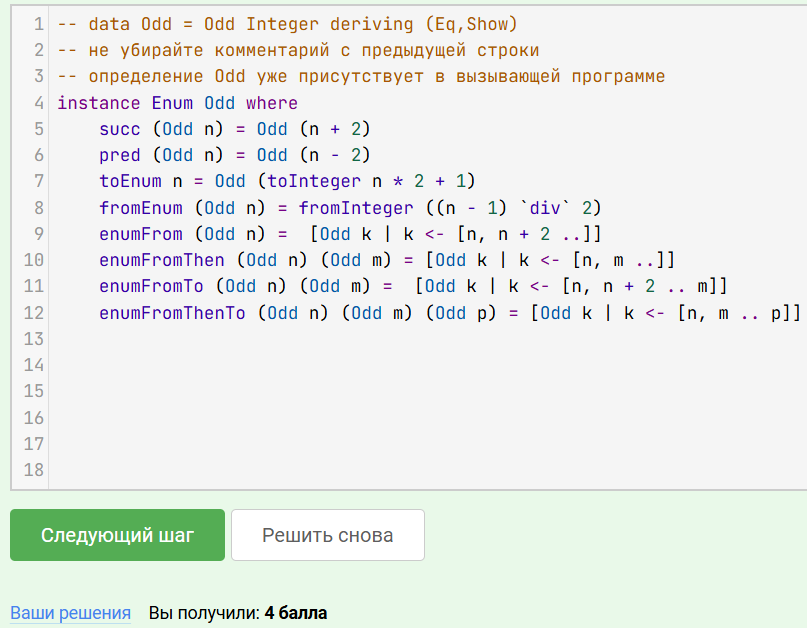
Odd 33

и использовать конструктор данных Odd в сопоставлении с образцом:

addEven :: Odd -> Integer -> Odd

addEven (Odd n) m | m `mod` 2 == 0 = Odd (n + m)

| otherwise = error "addEven: second parameter cannot be odd"



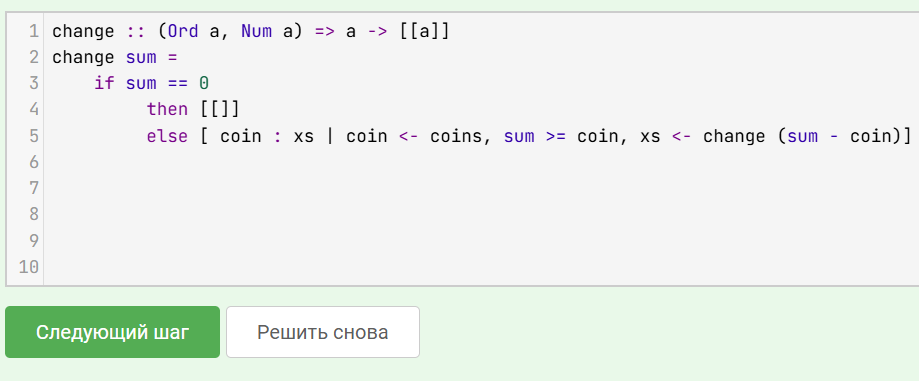
Пусть есть список положительных достоинств монет coins, отсортированный по возрастанию. Воспользовавшись механизмом генераторов списков, напишите функцию change, которая разбивает переданную ей положительную сумму денег на монеты достоинств из списка coins всеми возможными способами. Например, если coins = [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.

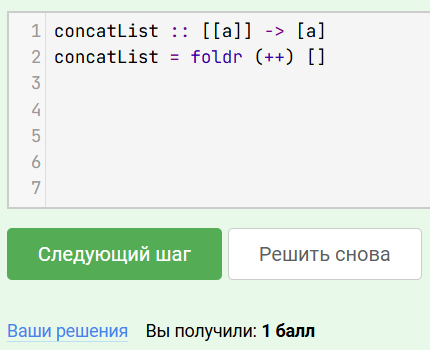
Список coins определять не надо.



Напишите реализацию функции concatList через foldr

GHCi> concatList [[1,2],[],[3]]

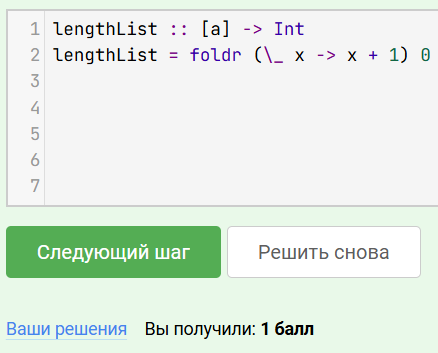
[1,2,3]



Используя функцию foldr, напишите реализацию функции lengthList, вычисляющей количество элементов в списке.

GHCi> lengthList [7,6,5]

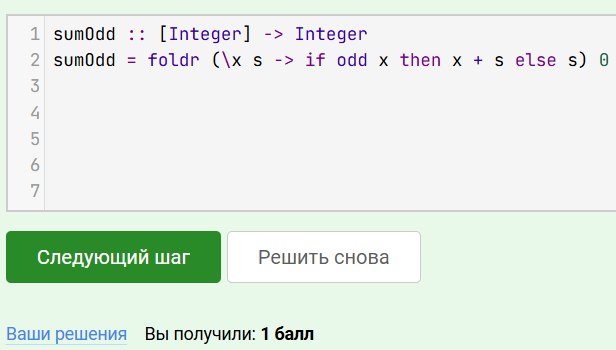
3



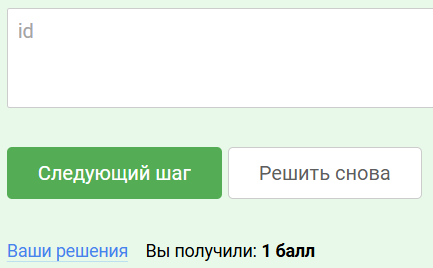
Реализуйте функцию sumOdd, которая суммирует элементы списка целых чисел, имеющие нечетные значения:

GHCi> sumOdd [2,5,30,37]

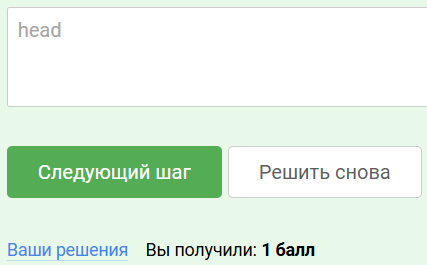
42



Какой функции стандартной библиотеки, суженной на списки, эквивалентно выражение foldr (:) []?



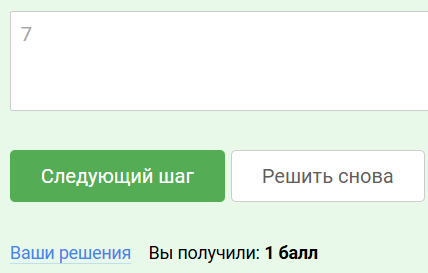
Какой функции стандартной библиотеки эквивалентно выражение foldr const undefined?



При каком значении переменной x следующие два выражения примут одно и то же значение (отличное от неопределенного)?

foldr (-) x [2,1,5]

foldl (-) x [2,1,5]

x

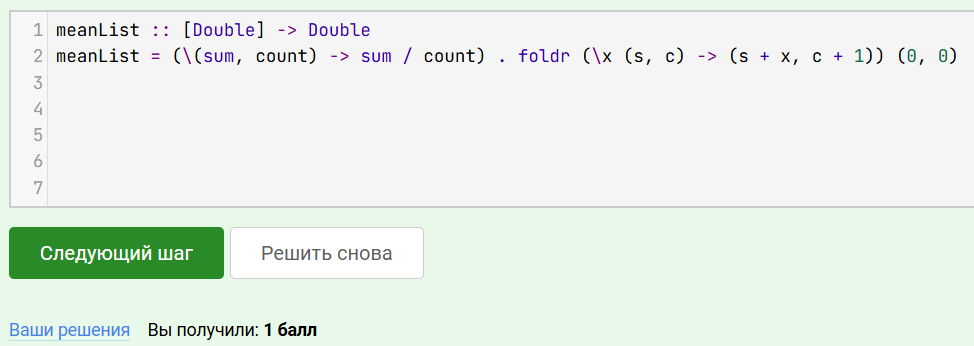
Реализуйте функцию meanList, которая находит среднее значение элементов списка, используя однократный вызов функции свертки.

GHCi> meanList [1,2,3,4]

2.5

Постобработка считается допустимой, то есть предполагаемая реализация функции meanList имеет вид

meanList = someFun . foldr someFoldingFun someIni



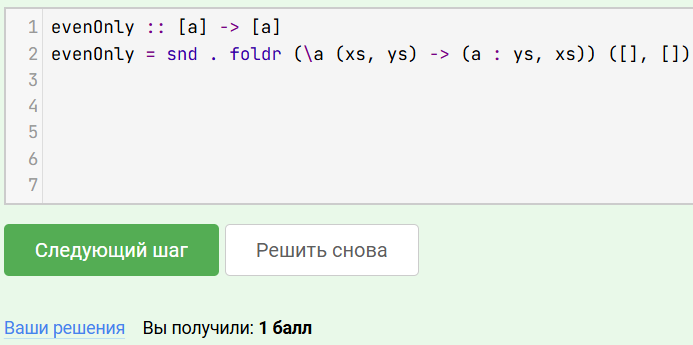
Используя однократный вызов свертки, реализуйте функцию evenOnly, которая выбрасывает из списка элементы, стоящие на нечетных местах, оставляя только четные.

GHCi> evenOnly [1..10]

[2,4,6,8,10]

GHCi> evenOnly ['a'..'z']

"bdfhjlnprtvxz"



Попробуйте добиться того, чтобы реализованная вами в прошлом задании функция evenOnly позволяла работать и с бесконечными списками.

То есть, например, запрос на первые три элемента бесконечного списка, возвращаемого этой функцией, примененной к списку всех натуральных чисел, должен завершаться:

GHCi> take 3 (evenOnly [1..])

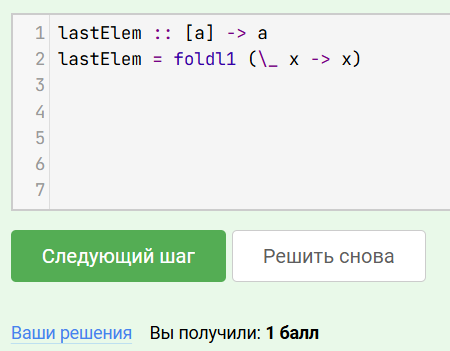
[2,4,6]



Напишите реализацию функции, возвращающей последний элемент списка, через foldl1.

lastElem :: [a] -> a

lastElem = foldl1 undefined



Используя unfoldr, реализуйте функцию, которая возвращает в обратном алфавитном порядке список символов, попадающих в заданный парой диапазон. Попадание символа x в диапазон пары (a,b) означает, что x >= a и x <= b.

revRange :: (Char,Char) -> [Char]

revRange = unfoldr g

where g = undefined

GHCi> revRange ('a','z')

"zyxwvutsrqponmlkjihgfedcba"

