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

**Модуль 1: Введение**

***Раздел 1.1 Установка и настройка***

**Шаг 6**

Запустите ваш текстовой редактор и создайте файл Hello.hs, содержащий следующую строку кода:

main = putStrLn "Hello, world!"

Вызовите теперь с помощью средств вашей ОС интерпретатор GHCi c параметром — именем файла исходного кода:

ghci Hello.hs

(Файл должен располагаться в том же каталоге, откуда происходит вызов интерпретатора.) Проверьте, что загрузка модуля прошла успешно, вызвав в интерпретаторе определенную вами функцию main:

GHCi> main

Hello, world!

Какое приглашение на самом деле выдает командная строка интерпретатора (в предыдущем примере интерпретатор выдал приглашение GHCi> )?

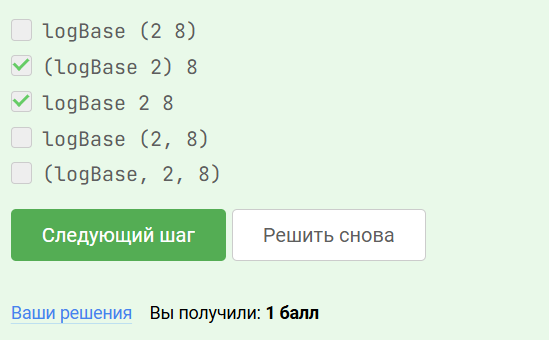
**Ответ:** \*Main>

***Раздел 1.2 Функции***

**Шаг 5**

В стандартной библиотеке Haskell есть функция вычисления логарифма по произвольному основанию logBase. Это функция двух переменных, которой требуется передать основание логарифма и аргумент, на котором логарифм будет вычислен. Какие из следующих вызовов обеспечат вычисление логарифма по основанию 2 от 8?

**Решение – результат:**



***Раздел 1.2 Функции***

**Шаг 7**

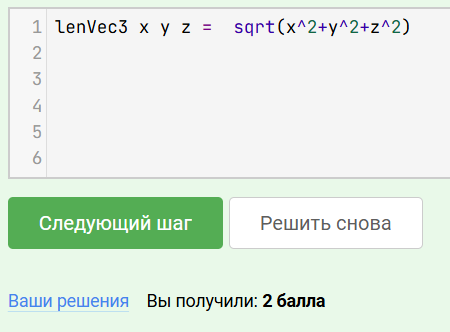
Реализуйте функцию трех аргументов lenVec3, которая вычисляет длину трехмерного вектора. Аргументы функции задают декартовы координаты конца вектора, его начало подразумевается находящимся в начале координат. Для извлечения квадратного корня воспользуйтесь функцией sqrt, определенной в стандартной библиотеке.

GHCi> lenVec3 2 3 6  
7.0

Для возведения в степень используем ^, для извлечение квадрата используем sqrt, определенной в стандартной библиотеке.

Сумма квадратов координат вычисляется следующим образом: x^2 + y^2 + z^2. Из суммы квадратов координат извлекается квадратный корень.

**Решение – результат:**



***Раздел 1.2 Функции***

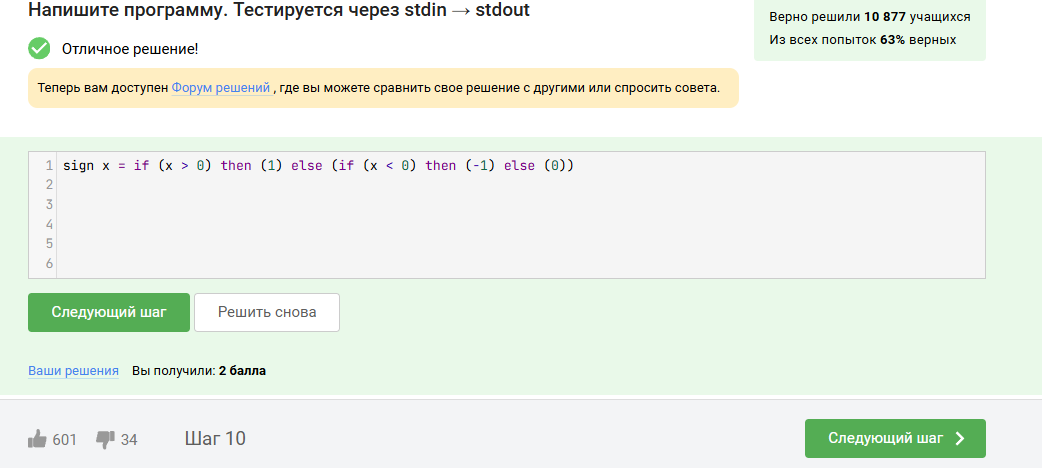
**Шаг 10**

Напишите реализацию функции sign, которая возвращает 1, если ей передано положительное число, (-1), если отрицательное, и 0 в случае, когда передан 0.

GHCi> sign (-100)

-1  
Для решения используют if then else

**Решение – результат:**

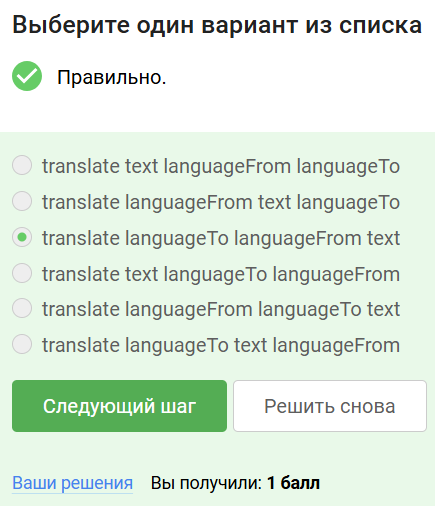


***Раздел 1.2 Функции***

**Шаг 12**

Предположим, мы разрабатываем на Haskell интерфейс системы перевода для естественных языков. Он должен содержать функцию translate с параметрами text, languageFrom и languageTo. Расположите параметры в таком порядке, чтобы было удобно реализовывать следующие функции: translateFromSpanishToRussian, translateFromEnglishToRussian и translateToRussian.

**Решение – результат:**



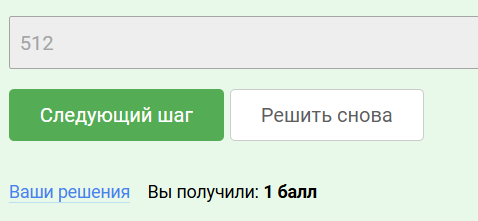
***Раздел 1.3 Операторы***

**Шаг 4**

Попробуйте вычислить значение выражения 2 ^ 3 ^ 2, не используя GHCi.

Правостороннее у ^.

**Решение – результат:**

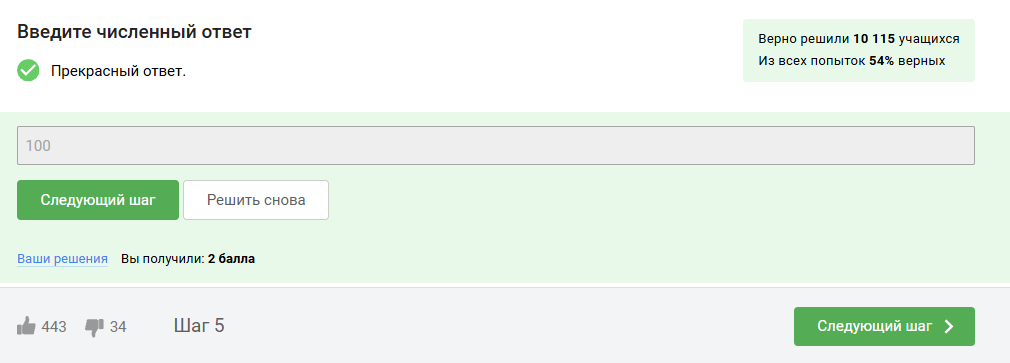


***Раздел 1.3 Операторы***

**Шаг 5**

Попробуйте вычислить значение выражения (\*) 2 ((+) 1 4) ^ 2, не используя GHCi.

**Решение – результат:**



***Раздел 1.3 Операторы***

**Шаг 7**

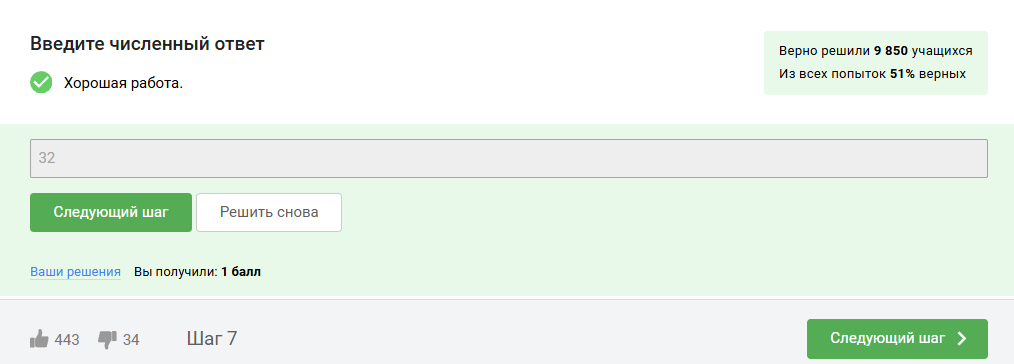
Используя данное выше определение оператора (\*+\*):

infixl 6 \*+\*

(\*+\*) a b = a ^ 2 + b ^ 2

попробуйте устно вычислить значение выражения 1 + 3 \*+\* 2 \* 2.

**Решение – результат:**



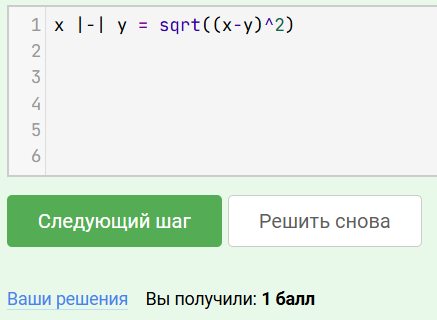
***Раздел 1.3 Операторы***

**Шаг 8**

Реализуйте оператор |-|, который возвращает модуль разности переданных ему аргументов:

GHCi> 5 |-| 72

**Решение – результат:**

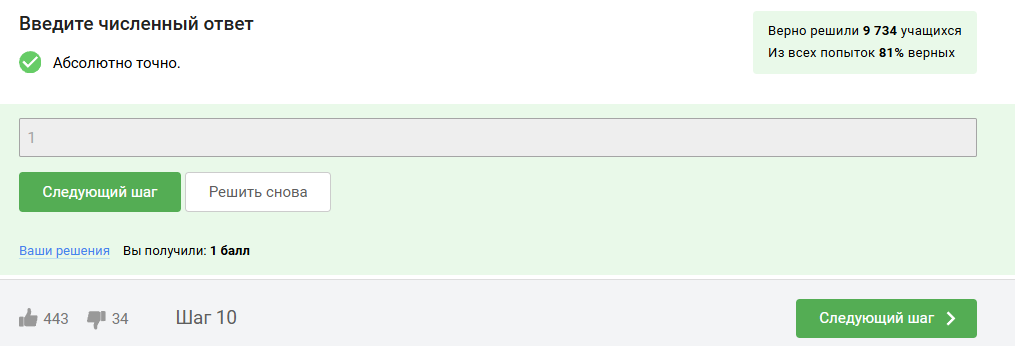


***Раздел 1.3 Операторы***

**Шаг 10**

Попробуйте вычислить значение выражения (`mod` 14) ((+ 5) 10), не используя GHCi. (Функция mod возвращает остаток от целочисленного деления первого своего аргумента на второй.)

**Решение – результат:**

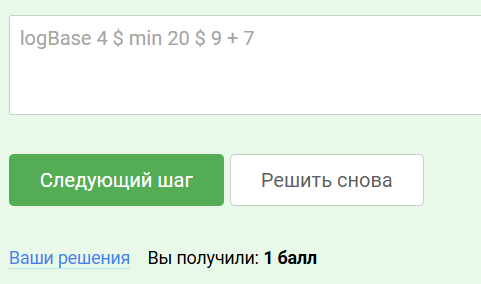


***Раздел 1.3 Операторы***

**Шаг 12**

Используя оператор $, перепишите выражение logBase 4 (min 20 (9 + 7)) без скобок. (Разделяйте все токены одним пробелом.)

**Решение – результат:**

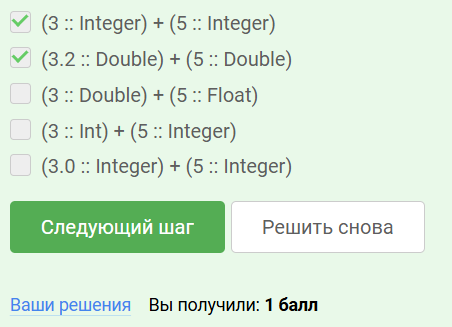


***Раздел 1.4 Базовые типы***

**Шаг 4**

Какие из следующих выражений типизированы верно, то есть не приводят к ошибкам типа?

**Решение – результат:**



***Раздел 1.4 Базовые типы***

**Шаг 6**

Вспомним функцию discount, которая возвращала итоговую сумму покупки с возможной скидкой. В качестве параметров ей передавались сумма без скидки sum, процент скидки proc, причем скидка начислялась, если переданная сумма превышает порог limit. Все эти параметры, как и возвращаемое значение, можно хранить в типе Double. (Здесь следует отметить, что в реальных финансовых приложениях использовать тип с плавающей точкой для хранения подобной информации не рекомендуется.) Тип функции можно задать в файле исходного кода вместе с ее определением:

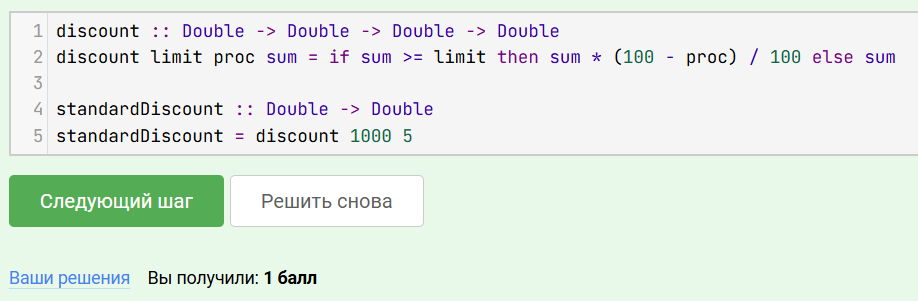
discount :: Double -> Double -> Double -> Doublediscount limit proc sum = if sum >= limit then sum \* (100 - proc) / 100 else sum

Отметим, что объявление типа необязательно, хотя часто рекомендуется в качестве документации. Его обычно располагают перед определением функции, хотя это объявление верхнего уровня можно расположить в любом месте файла с исходным кодом.

Запишите тип функции standardDiscount, определенной как частичное применение функции discount:

standardDiscount :: ???standardDiscount = discount 1000 5

**Решение – результат:**

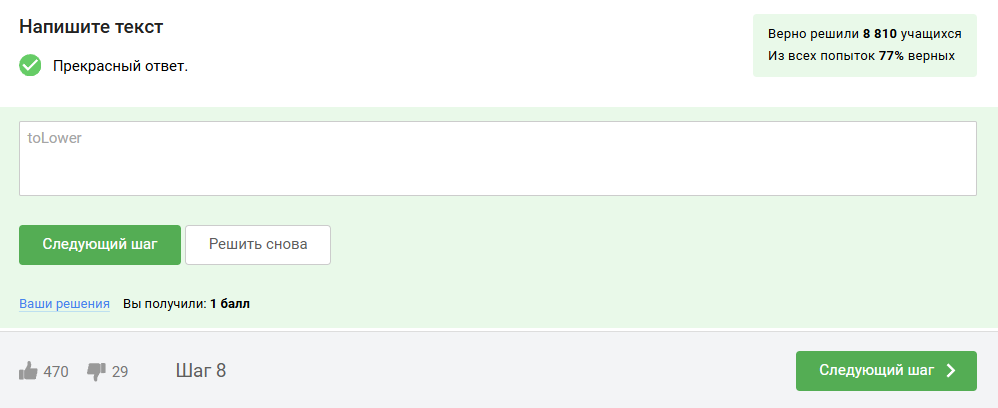


***Раздел 1.4 Базовые типы***

**Шаг 8**

Воспользовавшись справочной системой [Hoogle](https://www.haskell.org/hoogle/), найдите имя функции типа Char -> Char, переводящей символ в нижний регистр.

**Решение – результат:**



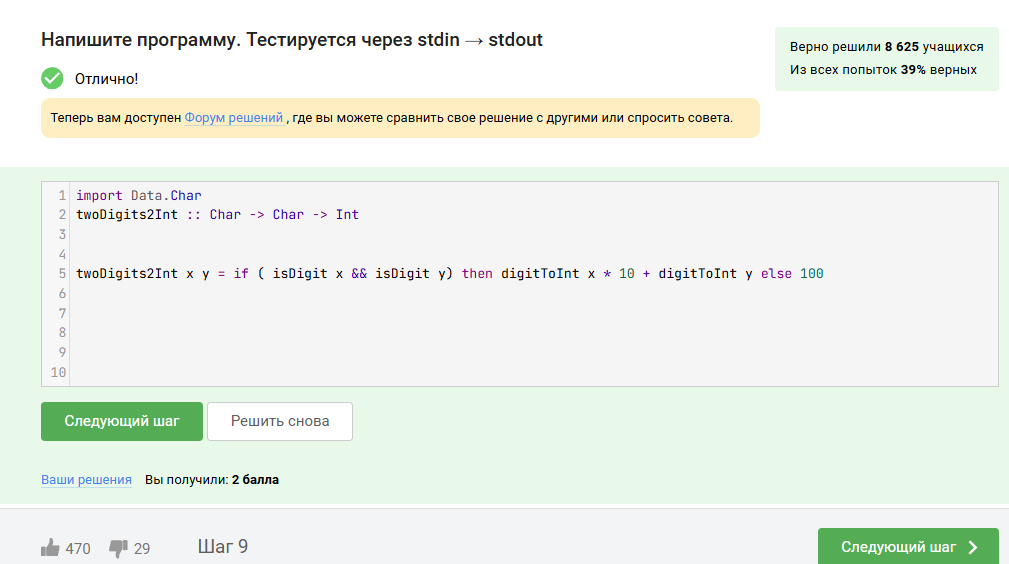
***Раздел 1.4 Базовые типы***

**Шаг 9**

Реализуйте функцию twoDigits2Int, которая принимает два символа и возвращает число, составленное из этих символов, если оба символа числовые, и 100 в противном случае. (Первый символ рассматривается как количество десятков, второй — единиц.)

GHCi> twoDigits2Int '4' '2'42

**Решение – результат:**



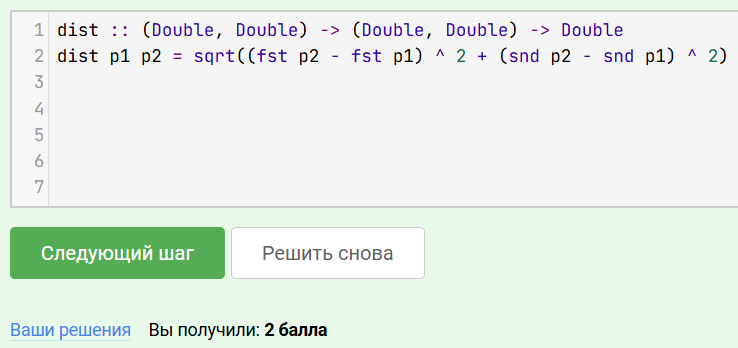
***Раздел 1.4 Базовые типы***

**Шаг 11**

Будем задавать точки на плоскости парами типа (Double, Double). Реализуйте функцию dist, которая возвращает расстояние между двумя точками, передаваемыми ей в качестве аргументов.

dist :: (Double, Double) -> (Double, Double) -> Doubledist p1 p2 = ???

**Решение – результат:**

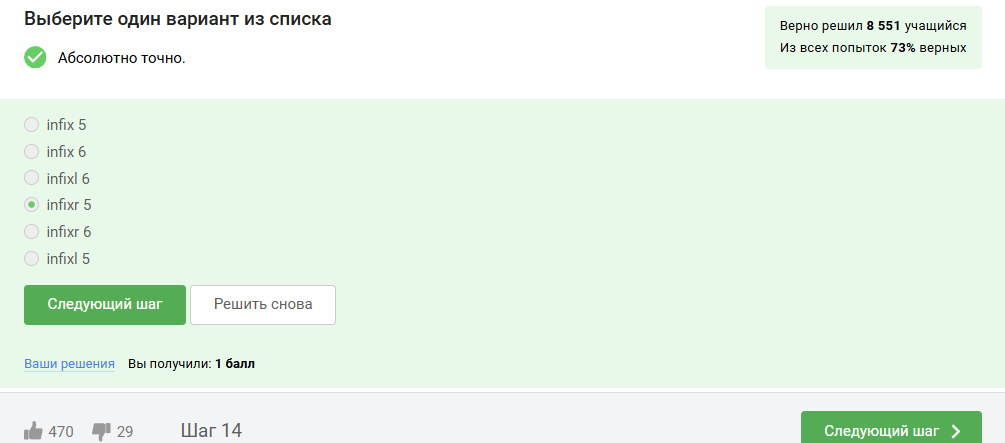


***Раздел 1.4 Базовые типы***

**Шаг 14**

Операторы (:) и (++) имеют одинаковую ассоциативность и приоритет. Укажите их. (Воспользуйтесь командой интерпретатора GHCi :info).

**Решение – результат:**

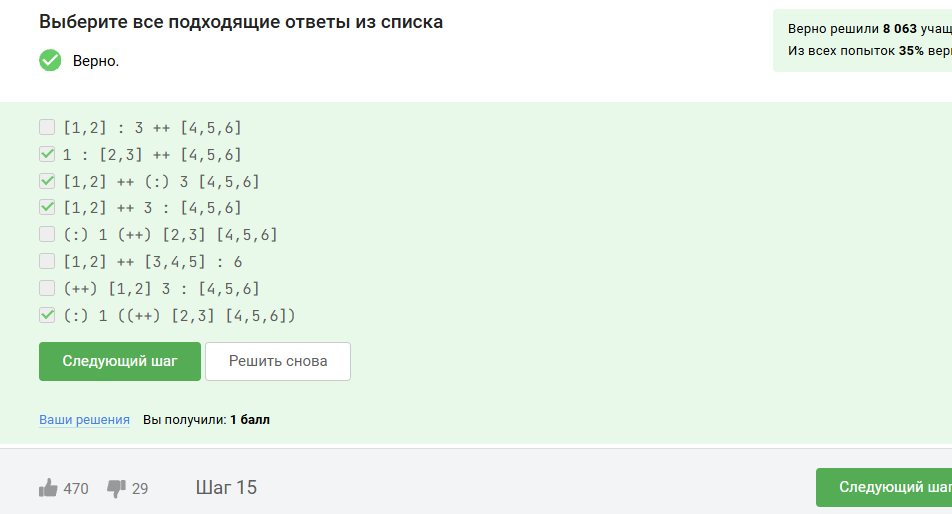


***Раздел 1.4 Базовые типы***

**Шаг 15**

Не используя GHCi, выберите выражения, проходящие проверку типов.

**Решение – результат:**

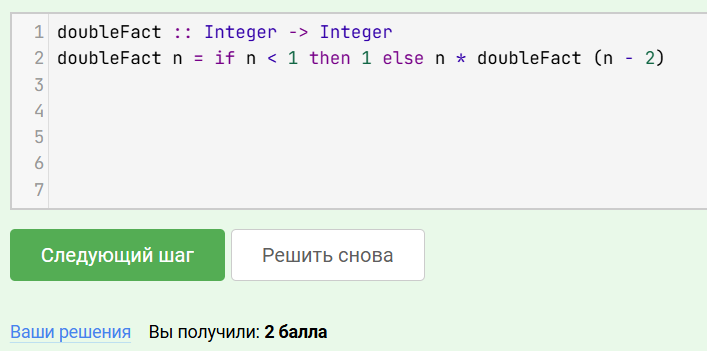


***Раздел 1.5 Рекурсия***

**Шаг 4**

Определите функцию, вычисляющую двойной факториал, то есть произведение натуральных чисел, не превосходящих заданного числа и имеющих ту же четность. Например: 7!!=7⋅5⋅3⋅17!!=7⋅5⋅3⋅1,  8!!=8⋅6⋅4⋅28!!=8⋅6⋅4⋅2. Предполагается, что аргумент функции может принимать только неотрицательные значения.

**Решение – результат:**



***Раздел 1.5 Рекурсия***

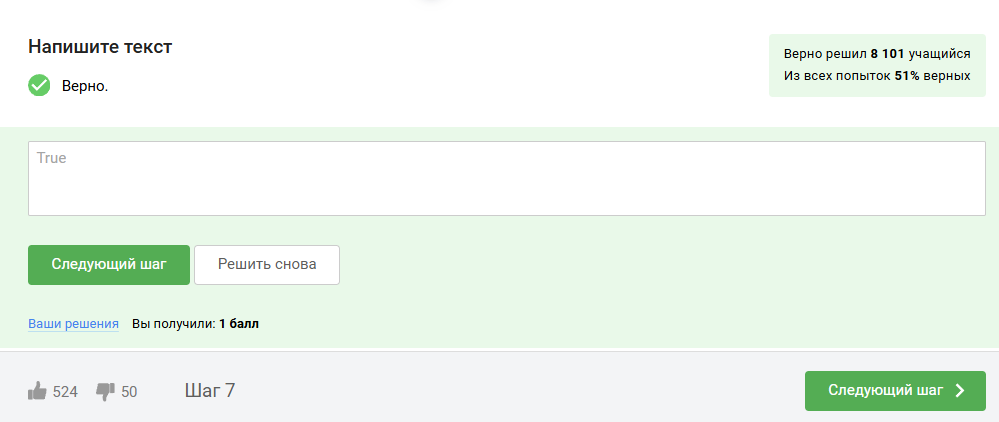
**Шаг 7**

В последнем примере предыдущего шага в охранном выражении использовался идентификатор otherwise. Это не ключевое слово, а константа, определенная для удобства в стандартной библиотеке:

otherwise = ?

Как вы думаете, какова правая часть её определения?

**Решение – результат:**



***Раздел 1.5 Рекурсия***

**Шаг 8**

Последовательность чисел Фибоначчи 0,1,1,2,3,5,8,13,21,…0,1,1,2,3,5,8,13,21,… легко определить рекурсивно, задав два первых терминирующих значения и определив любое последующее как сумму двух непосредственно предыдущих:

F0=0

F1=1

Fn=Fn−1+Fn−2​

На Haskell данное определение задаётся следующей функцией:

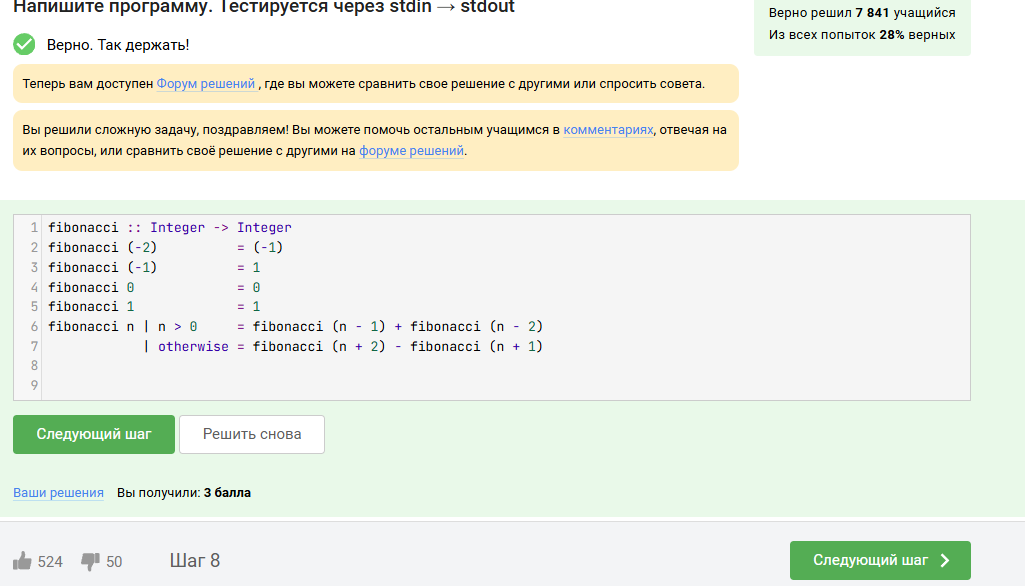
fibonacci 0 = 0fibonacci 1 = 1fibonacci n = fibonacci (n - 1) + fibonacci (n - 2)

Эта функция определена лишь для неотрицательных чисел. Однако, из данного выше определения можно вывести формулу для вычисления чисел Фибоначчи при отрицательных индексах, при этом последовательность будет следующей:

F(−1)=1,F(−2)=−1,…,F(−10)=−55

Измените определение функции fibonacci так, чтобы она была определена для всех целых чисел и порождала при отрицательных аргументах указанную последовательность.

**Решение – результат:**



***Раздел 1.5 Рекурсия***

**Шаг 10**

Реализация функции для вычисления числа Фибоначчи, основанная на прямом рекурсивном определении, крайне неэффективна - количество вызовов функции растет экспоненциально с ростом значения аргумента. GHCi позволяет отслеживать использование памяти и затраты времени на вычисление выражения, для этого следует выполнить команду :set +s:

GHCi> :set +sGHCi> fibonacci 30832040

(8.36 secs, 298293400 bytes)

С помощью механизма аккумуляторов попробуйте написать более эффективную реализацию, имеющую линейную сложность (по числу рекурсивных вызовов). Как и в предыдущем задании, функция должна быть определена для всех целых чисел.

**Решение – результат:**

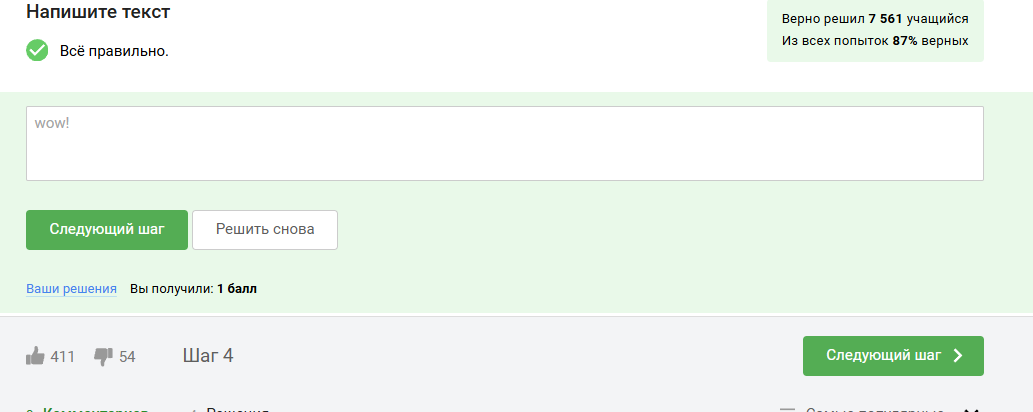


***Раздел 1.6 Локальные связывания и правила отступов***

**Шаг 4**

Не используя GHCi, определите строку, которая является значением выражения (let x = 'w' in [x,'o',x]) ++ "!".

**Решение – результат:**



***Раздел 1.6 Локальные связывания и правила отступов***

**Шаг 6**

Реализуйте функцию seqA, находящую элементы следующей рекуррентной последовательности

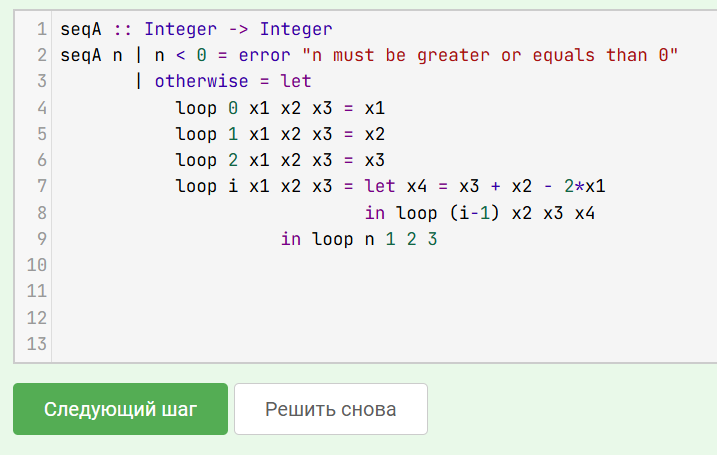
a(0)=1;a(1)=2;a(2)=3;a(k+3)=a(k+2)+a(k+1)−2a(k)

Попытайтесь найти эффективное решение.

GHCi> seqA 301

1276538859311178639666612897162414

**Решение – результат:**

. 

***Раздел 1.6 Локальные связывания и правила отступов***

**Шаг 8**

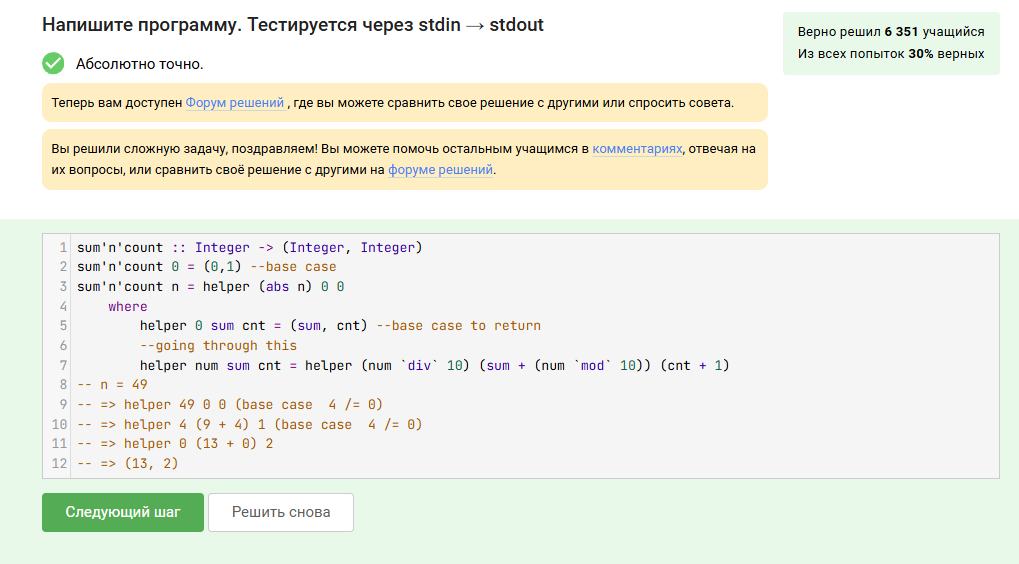
Реализуйте функцию, находящую сумму и количество цифр десятичной записи заданного целого числа.

sum'n'count :: Integer -> (Integer, Integer)sum'n'count x = undefined

GHCi> sum'n'count (-39)

(12,2)

**Решение – результат:**



***Раздел 1.6 Локальные связывания и правила отступов***

**Шаг 9**

Реализуйте функцию, находящую значение определённого интеграла от заданной функции ff на заданном интервале [a,b][a,b] методом трапеций. (Используйте равномерную сетку; достаточно 1000 элементарных отрезков.)

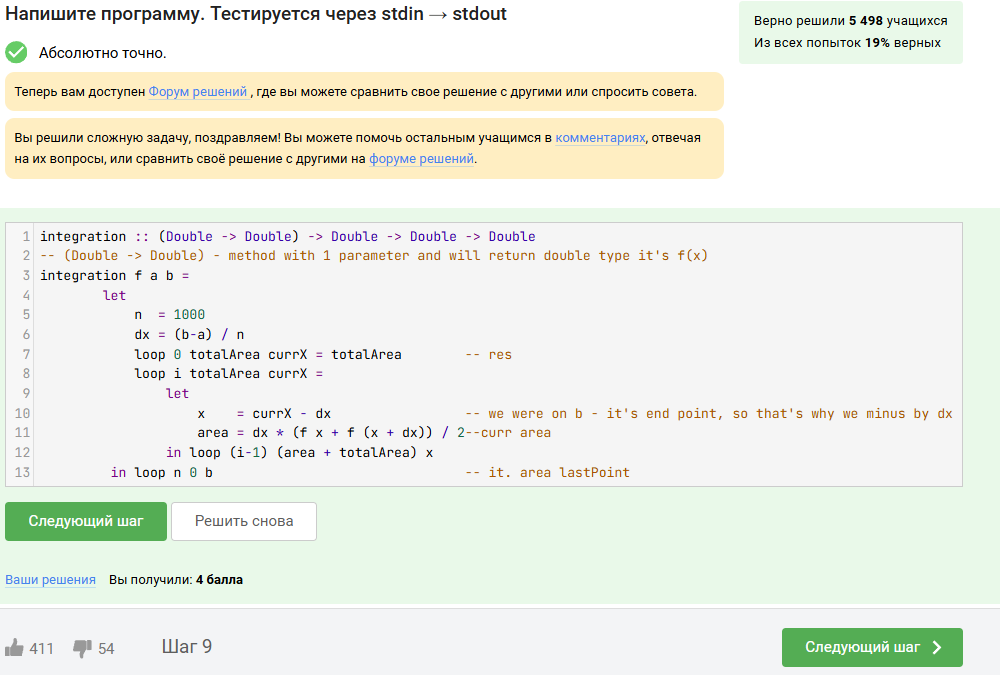
integration :: (Double -> Double) -> Double -> Double -> Doubleintegration f a b = undefined

GHCi> integration sin pi 0

-2.0

Результат может отличаться от -2.0, но не более чем на 1e-4.

**Решение – результат:**



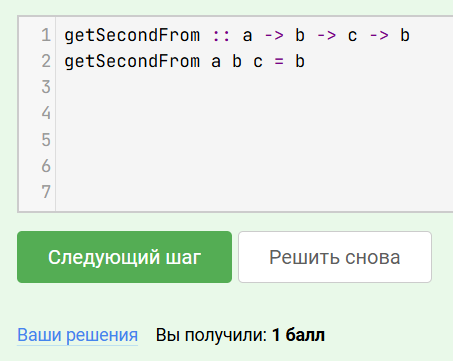
***Раздел 2.1 Параметрический полиморфизм***

**Шаг 3**

Напишите функцию трех аргументов getSecondFrom, полиморфную по каждому из них, которая полностью игнорирует первый и третий аргумент, а возвращает второй. Укажите ее тип.

GHCi> getSecondFrom True 'x' "Hello"'x'GHCi> getSecondFrom 'x' 42 True 42

**Решение – результат:**



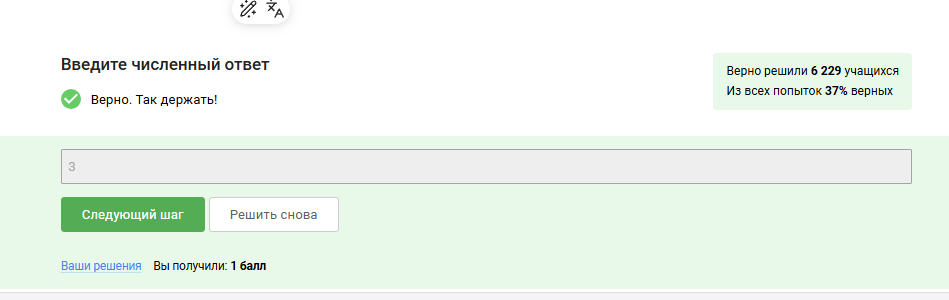
***Раздел 2.1 Параметрический полиморфизм***

**Шаг 4**

Сколько разных всегда завершающихся функций с типом a -> a -> b -> a -> a можно реализовать?

Две функции одинаковой арности считаются разными, если существует набор значений их аргументов, на котором они дают разные результирующие значения.

**Решение – результат:**



***Раздел 2.1 Параметрический полиморфизм***

**Шаг 7**

 модуле Data.Function определена полезная функция высшего порядка

on :: (b -> b -> c) -> (a -> b) -> a -> a -> con op f x y = f x `op` f y

Она принимает четыре аргумента: бинарный оператор с однотипными аргументами (типа b), функцию f :: a -> b, возвращающую значение типа b, и два значения типа a. Функция on применяет f дважды к двум значениям типа a и передает результат в бинарный оператор.

Используя on можно, например, записать функцию суммирования квадратов аргументов так:

sumSquares = (+) `on` (^2)

Функция multSecond, перемножающая вторые элементы пар, реализована следующим образом

multSecond = g `on` h

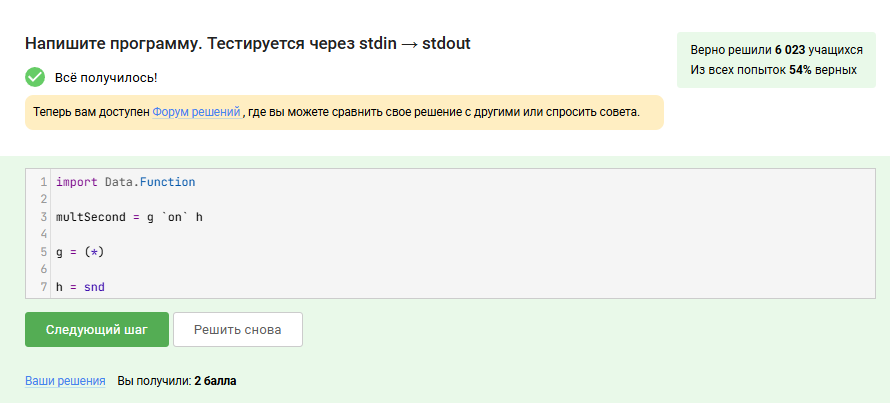
g = undefined

h = undefined

Напишите реализацию функций g и h.

GHCi> multSecond ('A',2) ('E',7)14

**Решение – результат:**



***Раздел 2.1 Параметрический полиморфизм***

**Шаг 9**

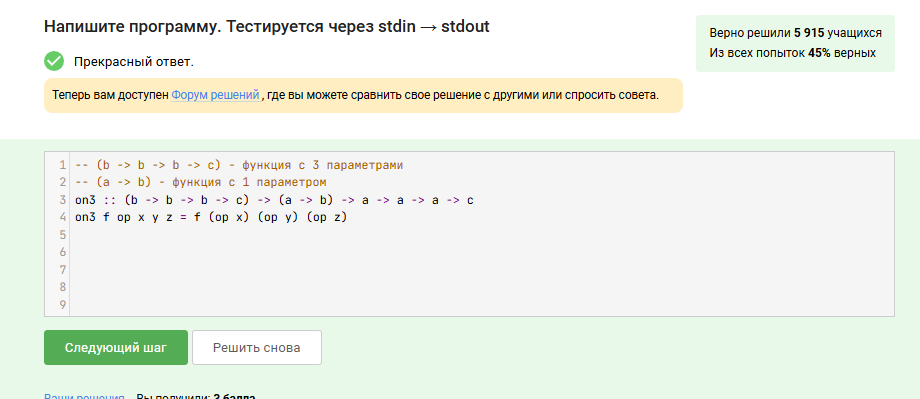
Реализуйте функцию on3, имеющую семантику, схожую с on, но принимающую в качестве первого аргумента трехместную функцию:

on3 :: (b -> b -> b -> c) -> (a -> b) -> a -> a -> a -> con3 op f x y z = undefined

Например, сумма квадратов трех чисел может быть записана с использованием on3 так

GHCi> let sum3squares = (\x y z -> x+y+z) `on3` (^2)GHCi> sum3squares 1 2 314

**Решение – результат:**



***Раздел 2.2 Параметрический полиморфизм (2)***

**Шаг 3**

Функция одной переменной doItYourself выбирает наибольшее из переданного ей аргумента и числа 42, затем возводит результат выбора в куб и, наконец, вычисляет логарифм по основанию 2 от полученного числа. Эта функция реализована в виде:

doItYourself = f . g . h

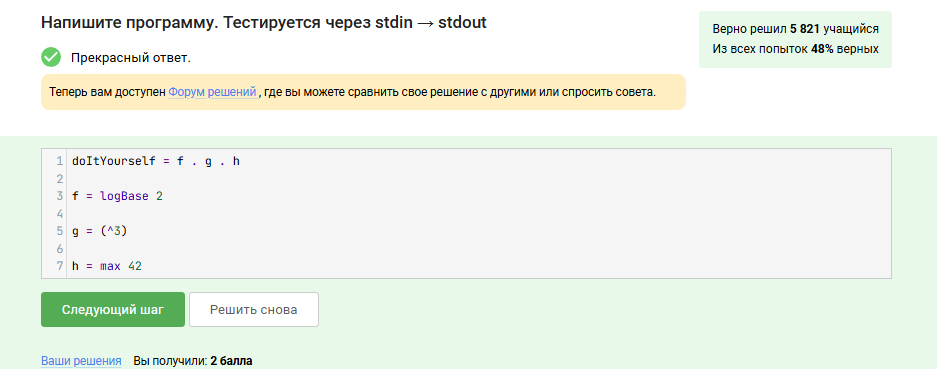
Напишите реализации функций f, g и h. Постарайтесь сделать это в бесточечном стиле.

f = undefined

g = undefined

h = undefined

**Решение – результат:**

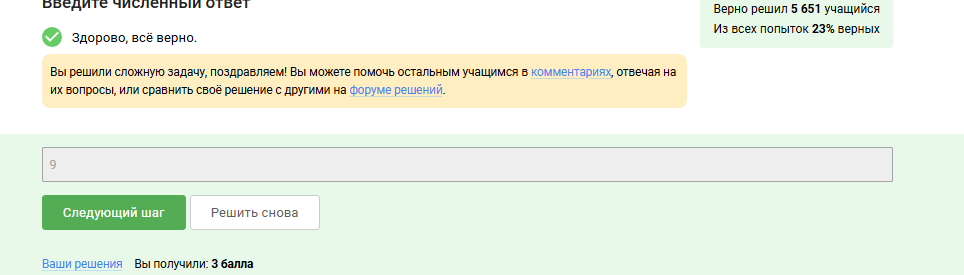


***Раздел 2.2 Параметрический полиморфизм (2)***

**Шаг 5**

Сколько разных всегда завершающихся функций с типом a -> (a,b) -> a -> (b,a,a) можно реализовать?

**Решение – результат:**

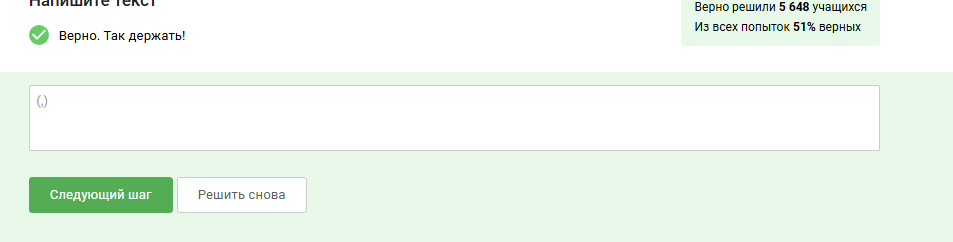


***Раздел 2.2 Параметрический полиморфизм (2)***

**Шаг 7**

Какому известному вам библиотечному оператору, конструктору или функции эквивалентно выражение curry id?

**Решение – результат:**

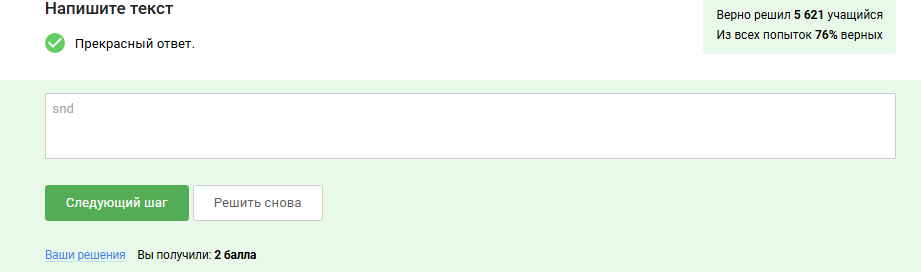


***Раздел 2.2 Параметрический полиморфизм (2)***

**Шаг 8**

Какому известному вам библиотечному оператору, конструктору или функции эквивалентно выражение uncurry (flip const)?

**Решение – результат:**



***Раздел 2.2 Параметрический полиморфизм (2)***

**Шаг 9**

В модуле Data.Tuple стандартной библиотеки определена функция swap :: (a,b) -> (b,a), переставляющая местами элементы пары:

GHCi> swap (1,'A')

('A',1)

Эта функция может быть выражена в виде:

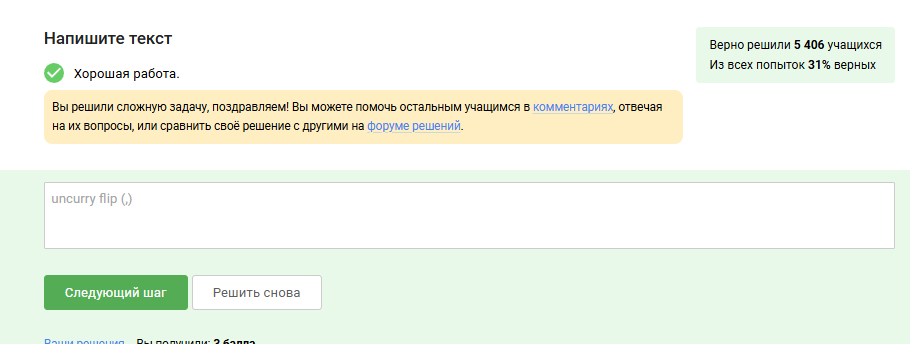
swap = f (g h)

где f, g и h — некоторые идентификаторы из следующего набора:

curry uncurry flip (,) const

Укажите через запятую подходящую тройку f,g,h.

**Решение – результат:**



***Раздел 2.3 Классы типов***

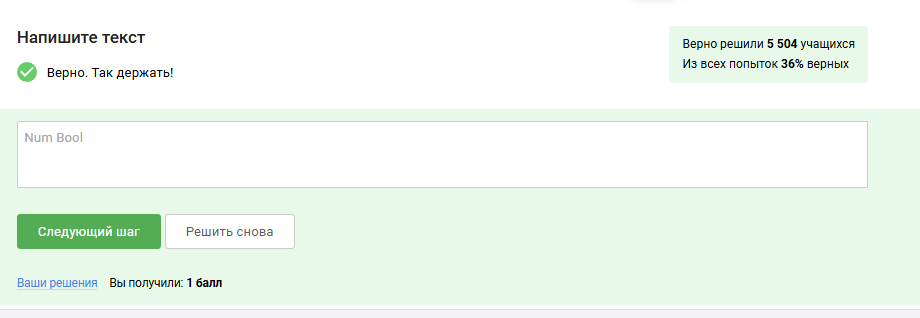
**Шаг 3**

На нехватку какого представителя какого класса типов пожалуется интерпретатор при попытке вывести тип выражения

True + False \* False

Запишите ответ в виде Имя\_класса\_типов Имя\_типа. Постарайтесь ответить, не используя GHCi.

**Решение – результат:**



***Раздел 2.3 Классы типов***

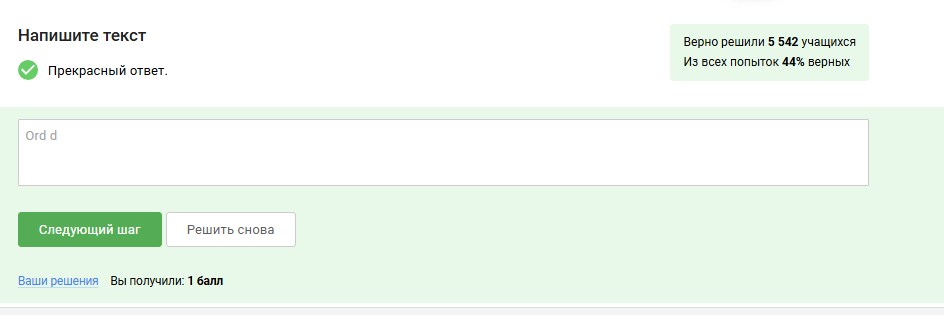
**Шаг 5**

Попробуйте, не используя GHCi или Hoogle, определить, какого контекста не хватает в типе функции

sort :: ? => [d] -> [d]

сортирующей переданный в нее список. Напишите выражение, которое должно стоять на месте знака вопроса.

**Решение – результат:**



***Раздел 2.3 Классы типов***

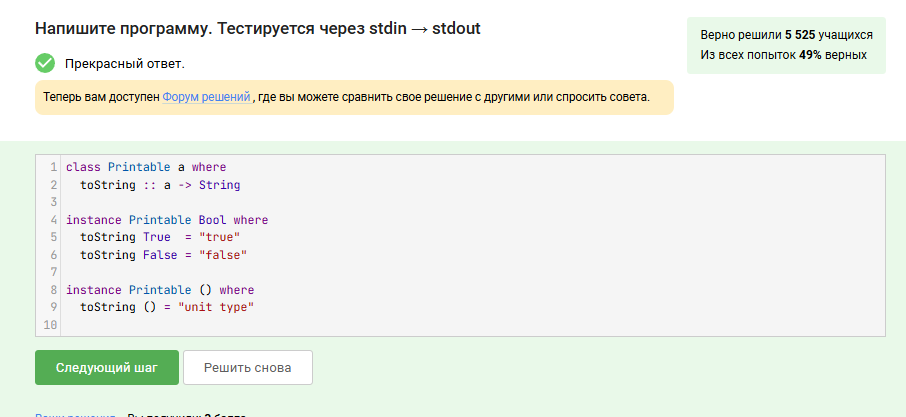
**Шаг 7**

Реализуйте класс типов Printable, предоставляющий один метод toString — функцию одной переменной, которая преобразует значение типа, являющегося представителем Printable, в строковое представление.

Сделайте типы данных Bool и () представителями этого класса типов, обеспечив следующее поведение:

GHCi> toString True"true"GHCi> toString False"false"GHCi> toString ()"unit type"

**Решение – результат:**



***Раздел 2.3 Классы типов***

**Шаг 9**

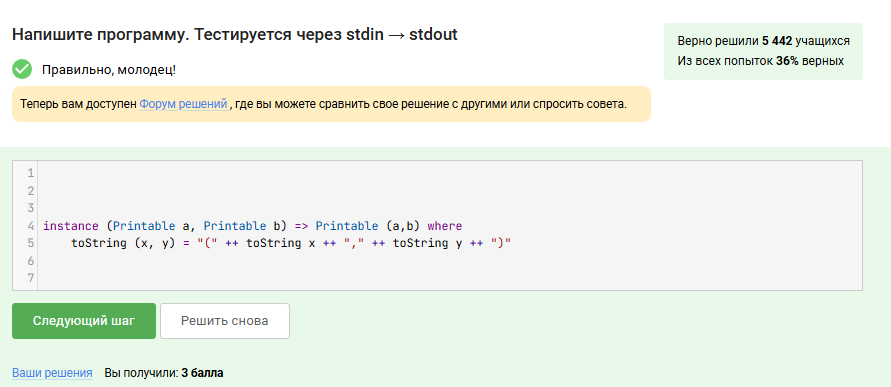
Сделайте тип пары представителем класса типов Printable, реализованного вами в предыдущей задаче, обеспечив следующее поведение:

GHCi> toString (False,())"(false,unit type)"

GHCi> toString (True,False)"(true,false)"

Примечание. Объявление класса типов Printable и представителей этого класса для типов () и  Bool заново реализовывать не надо — они присутствуют в программе, вызывающей ваш код.

**Решение – результат:**



***Раздел 2.4 Стандартные классы типов***

**Шаг 3**

Пусть существуют два класса типов KnownToGork и KnownToMork, которые предоставляют методы stomp (stab) и doesEnrageGork (doesEnrageMork) соответственно:

class KnownToGork a where

stomp :: a -> a

doesEnrageGork :: a -> Boolclass KnownToMork a where

stab :: a -> a

doesEnrageMork :: a -> Bool

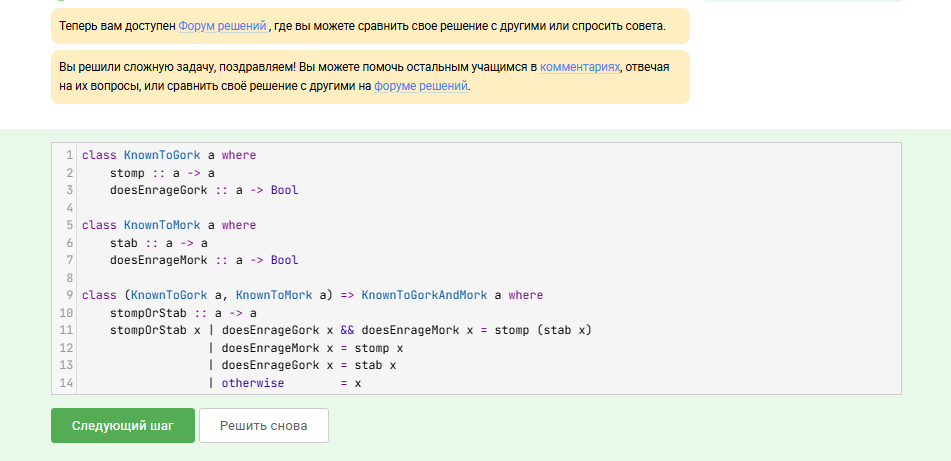
Класса типов KnownToGorkAndMork является расширением обоих этих классов, предоставляя дополнительно метод stompOrStab:

class (KnownToGork a, KnownToMork a) => KnownToGorkAndMork a where

stompOrStab :: a -> a

Задайте реализацию по умолчанию метода stompOrStab, которая вызывает метод stomp, если переданное ему значение приводит в ярость Морка; вызывает stab, если оно приводит в ярость Горка и вызывает сначала stab, а потом stomp, если оно приводит в ярость их обоих. Если не происходит ничего из вышеперечисленного, метод должен возвращать переданный ему аргумент.

**Решение – результат:**



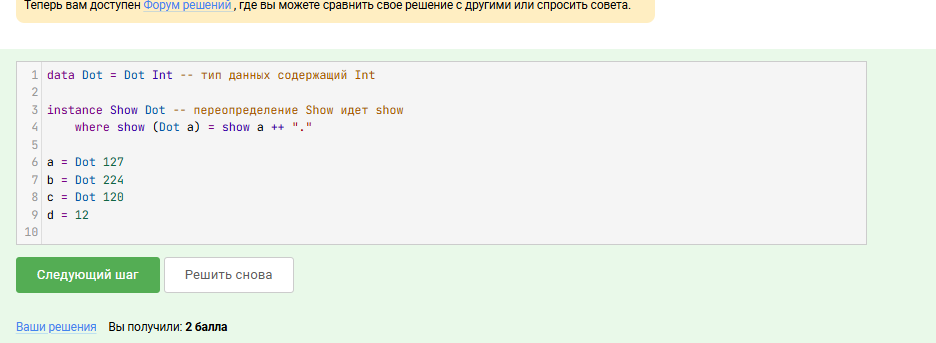
***Раздел 2.4 Стандартные классы типов***

**Шаг 5**

Имея функцию ip = show a ++ show b ++ show c ++ show d определите значения a, b, c, d так, чтобы добиться следующего поведения:

GHCi> ip"127.224.120.12"

**Решение – результат:**



***Раздел 2.4 Стандартные классы типов***

**Шаг 7**

Реализуйте класс типов

class SafeEnum a where

ssucc :: a -> a

spred :: a -> a

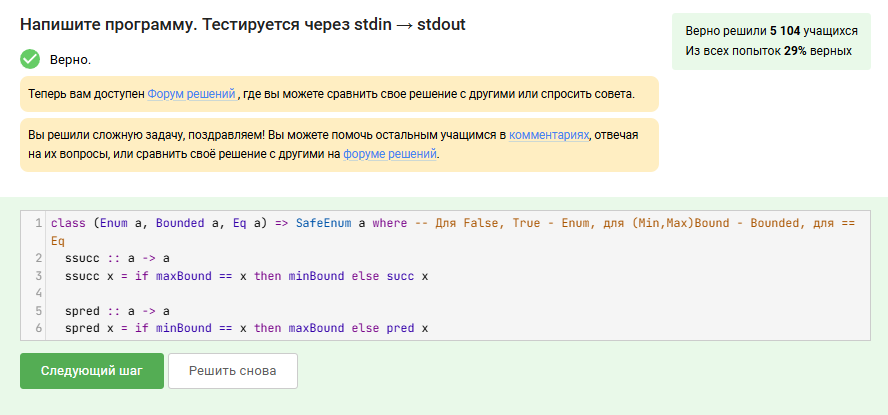
обе функции которого ведут себя как succ и pred стандартного класса Enum, однако являются тотальными, то есть не останавливаются с ошибкой на наибольшем и наименьшем значениях типа-перечисления соответственно, а обеспечивают циклическое поведение. Ваш класс должен быть расширением ряда классов типов стандартной библиотеки, так чтобы можно было написать реализацию по умолчанию его методов, позволяющую объявлять его представителей без необходимости писать какой бы то ни было код. Например, для типа Bool должно быть достаточно написать строку

instance SafeEnum Bool

и получить возможность вызывать

GHCi> ssucc FalseTrueGHCi> ssucc TrueFalse

**Решение – результат:**



***Раздел 2.4 Стандартные классы типов***

**Шаг 9**

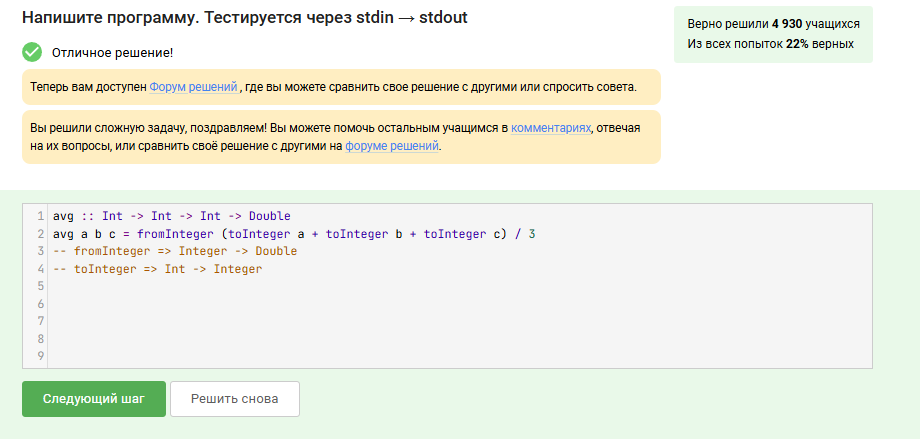
Напишите функцию с сигнатурой:

avg :: Int -> Int -> Int -> Double

вычисляющую среднее значение переданных в нее аргументов:

GHCi> avg 3 4 85.0

**Решение – результат:**



***Раздел 2.5 Нестрогая семантика***

**Шаг 3**

Предположим, что стандартные функции определены следующим образом:

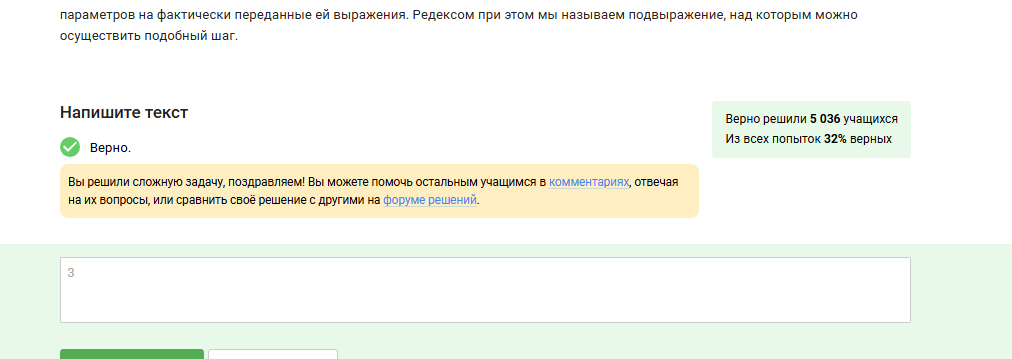
id x = xconst x y = xmax x y = if x <= y then y else xinfixr 0 $f $ x = f x

Сколько редексов имеется в следующем выражении

const $ const (4 + 5) $ max 42

Примечание. Мы определили шаг вычислений как подстановку тела функции вместо ее имени с заменой всех ее формальных параметров на фактически переданные ей выражения. Редексом при этом мы называем подвыражение, над которым можно осуществить подобный шаг.

**Решение – результат:**



***Раздел 2.5 Нестрогая семантика***

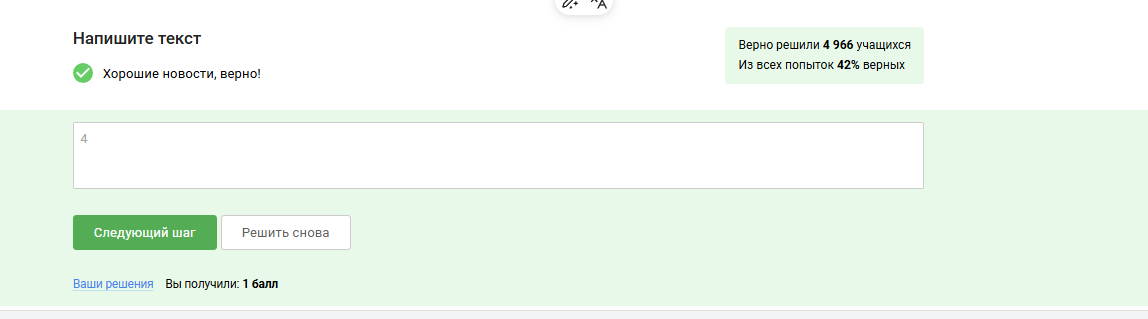
**Шаг 5**

Сколько шагов редукции потребуется, чтобы вычислить значение функции value, если используется ленивая стратегия вычислений с механизмом разделения?

bar x y z = x + yfoo a b = bar a a (a + b)value = foo (3 \* 10) (5 - 2)

Примечание. Подстановку тела функции value вместо value не считайте.

**Решение – результат:**



***Раздел 2.5 Нестрогая семантика***

**Шаг 7**

Отметьте функции, которые не могут привести к расходимости ни на каком корректном наборе аргументов.

foo a = a

bar = const foo

baz x = const True

quux = let x = x in x

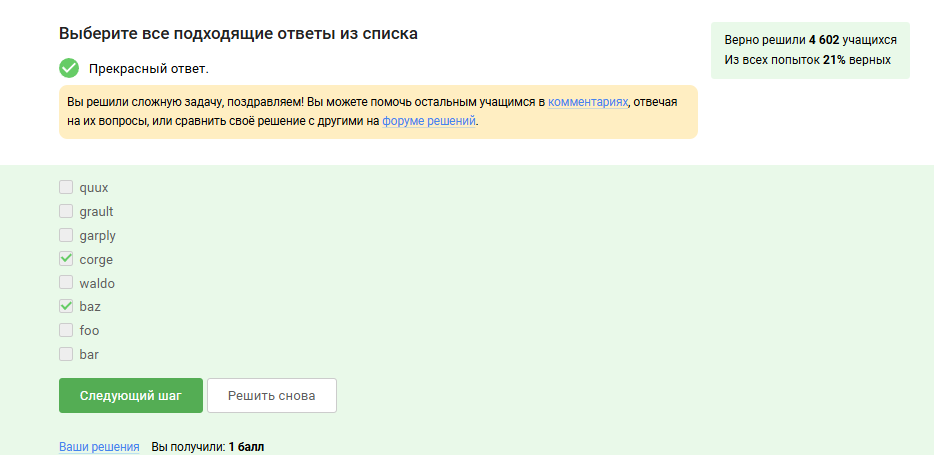
corge = "Sorry, my value was changed"

grault x 0 = xgrault x y = x

garply = grault 'q'

waldo = foo

**Решение – результат:**

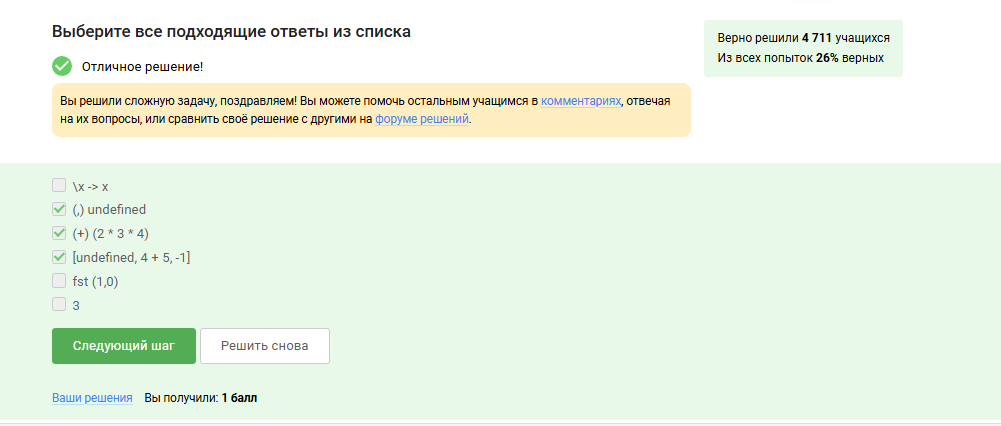


***Раздел 2.5 Нестрогая семантика***

**Шаг 9**

Какие из выражений ниже не находятся в нормальной форме, но находятся в слабой головной нормальной форме?

**Решение – результат:**



***Раздел 2.5 Нестрогая семантика***

**Шаг 11**

При вычислении каких из перечисленных ниже функций использование seq предотвратит нарастание количества невычисленных редексов при увеличении значения первого аргумента:

foo 0 x = xfoo n x = let x' = foo (n - 1) (x + 1)

in x' `seq` x'

bar 0 f = fbar x f = let f' = \a -> f (x + a)

x' = x - 1

in f' `seq` x' `seq` bar x' f'

baz 0 (x, y) = x + ybaz n (x, y) = let x' = x + 1

y' = y - 1

p = (x', y')

n' = n - 1

in p `seq` n' `seq` baz n' p

quux 0 (x, y) = x + yquux n (x, y) = let x' = x + 1

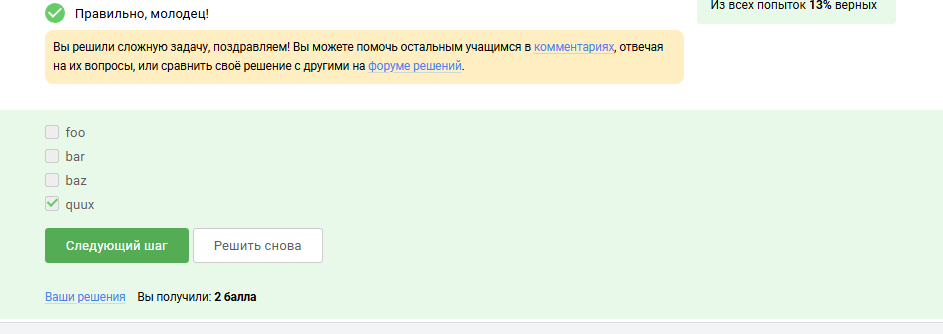
y' = y - 1

p = (x', y')

n' = n - 1

in x' `seq` y' `seq` n' `seq` quux n' p

**Решение – результат:**



***Раздел 2.5 Нестрогая семантика***

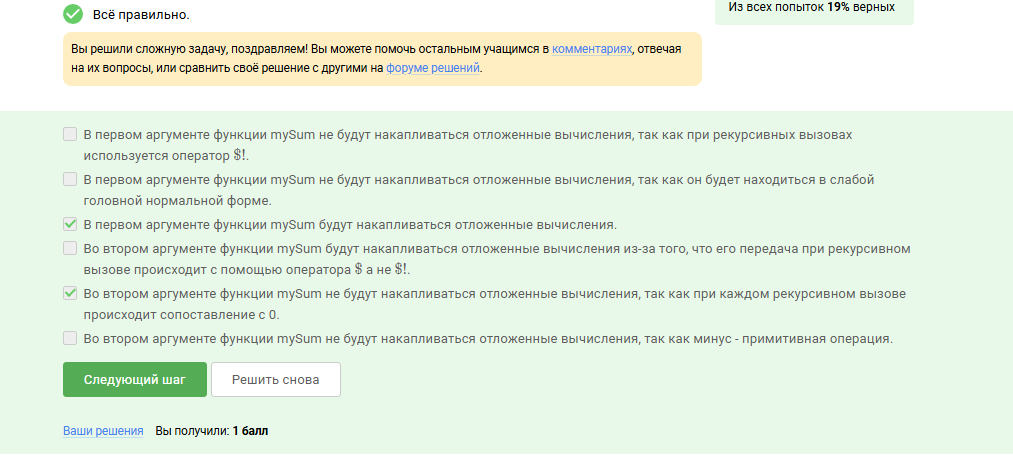
**Шаг 13**

Ниже определены функции mySum и goSum. Вызов goSum может выглядеть, к примеру, так:  goSum 15. Выберите верные утверждения, описывающие процесс вычисления подобного выражения.

mySum acc 0 = accmySum (result, ()) n = (mySum $! (result + n, ())) $ n - 1

goSum = mySum (0, ())

**Решение – результат:**



***Раздел 2.6 Модули и компиляция***

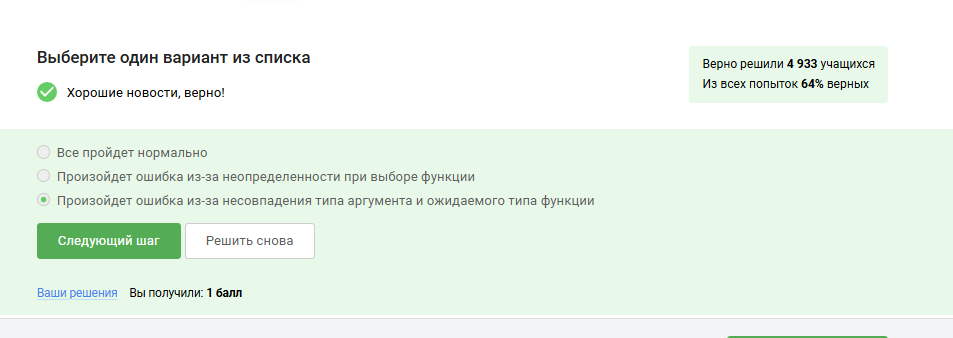
**Шаг 3**

Что произойдет при попытке загрузить данный модуль в GHCi?

module Test whereimport Data.List hiding (union)import Data.Set

myUnion [] ys = ysmyUnion xs ys = union xs ys

**Решение – результат:**



***Раздел 2.6 Модули и компиляция***

**Шаг 5**

Пусть модуль Foo содержит следующий код:

module Foo (a, b) where

a = undefinedb = undefinedc = undefined

а модуль Bar такой:

module Bar (a, d) where

import Foo (a, b)

d = undefined

Отметьте функции, доступные для использования после загрузки в модуле Baz со следующим кодом:

module Baz where

import Bar

**Решение – результат:**

