Самостоятельная работа студента в рамках курса «Функциональное программирование» («Haskell как первый язык программирования»)

Абрамов Сергей Михайлович*,
Герасин Тимофей Дмитриевич**,
Парменова Любовь Валерьевна*,
Порывай Максим Викторович**,
Юмагужин Николай Валерьевич*
* — Университет города Переславля имени А. К. Айламазяна,
** — филиал МГУ имени М. В. Ломоносова в городе Севастополе

27.09.2020

Автор решений задач

Студент: Иванов Иван Иванович.
Группа: 7М89.
Вуз: Университет города Переславля имени А. К. Айламазяна.
Дата начала решения задач: 20.09.2020.

Введение

Данный документ содержит литературный Haskell-код самостоятельной работы студента, выполненной в рамках курса «Функциональное программирование» («Haskell как первый язык программирования»).

Решения задач студент готовит и автономно отлаживает в системе GHCi. Для записи решений задач допускается и рекомендуется использование примитивов, определенных в прелюдии и в следующих двух стандартных модулях системы GHCi:

```
import Data.Char
import Data.List
```

Другие стандартные модулии системы GHCi использовать не разрешено.

Студентам строго рекомендовано изучить функции, определенные в данных модулях. Найти актуальное описание модулей легко, выполнив Google-поиск по запросам «Haskell Data.Char» и «Haskell Data.List». На найденных страницах с описанием модулей есть (в правом верхнем углу) ссылка на исходный код данных модулей, который полезно изучить. Как минимум, код тех функций, которые будут использоваться в решении задач. Особенно стоит обратить внимание на следующие функции:

```
(++) reverse sum product minimum concat iterate
repeat replicate take drop splitAt takeWhile
dropWhile span inits tails filter partition
elemIndices findIndices zipWith lines words unwords
nub delete union intersect sort sortOn sortBy
ord chr isDigit digitToInt isAlpha toUpper
```

Кроме того, используются типы, определенные в следующем модуле (подробнее см. в разделе 19 на странице 22):

```
import ExtraTypes
```

Предварительная проверка решений на тестах

Следующий текст предназначен для управления предварительной автоматической проверкой решений на наборах тестов. Сервис проверки доступен по адресу http://haskell.pereslavl.ru:

```
Для выполнения проверки некоторых (или всех) решений на сервере:
1) ниже в 1-й колонке соответствующих строк замените "-" на "+"
2) в третьей колонке этой строки напишите имя функции-решения
   данного задания (если использовали иное имя, а не указанное)
3) сохраните измененный файл
4) загрузите файл в форму автоматической проверки
5) дождитесь результата и изучите файл с результатом

   Проба пера
   - chk1_1     cube               -- проверка задачи 1-1
   - chk1_2     decmls             -- проверка задачи 1-2
   - chk1_3     ese                -- проверка задачи 1-3
   - chk1_4     eso                -- проверка задачи 1-4
   - chk1_5     epo                -- проверка задачи 1-5
   - chk1_6     n_cups_m_saucers   -- проверка задачи 1-6
   - chk1_7     wonderland3        -- проверка задачи 1-7
   - chk1_8     wonderland4        -- проверка задачи 1-8
   - chk1_9     fortee             -- проверка задачи 1-9
   - chk1_10    bwtable            -- проверка задачи 1-10
   - chk1_11    lottery            -- проверка задачи 1-11
   - chk1_12    ones3              -- проверка задачи 1-12
   - chk1_13    same3              -- проверка задачи 1-13
   - chk1_14    diff3              -- проверка задачи 1-14
   - chk1_15    gcd3               -- проверка задачи 1-15
```

```
- chk1_16    lcm2              -- проверка задачи 1-16
- chk1_17    lcm3              -- проверка задачи 1-17
- chk1_18    lcm3gcd           -- проверка задачи 1-18
- chk1_19    pps               -- проверка задачи 1-19

Корни уравнений
- chk2_1     abRoot            -- проверка задачи 2-1
- chk2_2     abcRoots          -- проверка задачи 2-2
- chk2_3     ab_Roots          -- проверка задачи 2-3
- chk2_4     abcdRoots         -- проверка задачи 2-4
- chk2_5     abcdeRoots        -- проверка задачи 2-5
- chk2_6     absRoots          -- проверка задачи 2-6
- chk2_7     abs2Roots         -- проверка задачи 2-7
- chk2_8     absabs2Roots      -- проверка задачи 2-8
- chk2_9     min_ab_Root       -- проверка задачи 2-9
- chk2_10    min_abcdRoot      -- проверка задачи 2-10
- chk2_11    min_abcdeRoot     -- проверка задачи 2-11
- chk2_12    min_absRoot       -- проверка задачи 2-12
- chk2_13    min_2absRoot      -- проверка задачи 2-13
- chk2_14    min_abs2absRoot   -- проверка задачи 2-14

Вычислительные задачи
- chk3_1     sumSq             -- проверка задачи 3-1
- chk3_2     fibs              -- проверка задачи 3-2
- chk3_3     bins              -- проверка задачи 3-3
- chk3_4     eApr              -- проверка задачи 3-4
- chk3_5     sqrtH             -- проверка задачи 3-5

Числа и цифры
- chk4_1     binary_dgts       -- проверка задачи 4-1
- chk4_2     ternary_dgts      -- проверка задачи 4-2
- chk4_3     decimal_dgts      -- проверка задачи 4-3
- chk4_4     k_ary_dgts        -- проверка задачи 4-4
- chk4_5     setun_dgts        -- проверка задачи 4-5
- chk4_6     s2dgts            -- проверка задачи 4-6
- chk4_7     p3dgts            -- проверка задачи 4-7
- chk4_8     l4dgts            -- проверка задачи 4-8

Строки и символы
- chk5_1     dgt_chr           -- проверка задачи 5-1
- chk5_2     chr_dgt           -- проверка задачи 5-2
- chk5_3     sumDigits         -- проверка задачи 5-3
- chk5_4     upAlpha           -- проверка задачи 5-4
- chk5_5     abbrev            -- проверка задачи 5-5

Кредит и его погашение
- chk6_1     balances          -- проверка задачи 6-1
- chk6_2     balance           -- проверка задачи 6-2
- chk6_3     credit_years      -- проверка задачи 6-3
- chk6_4     balances_y        -- проверка задачи 6-4
```

```
Делители и простые числа
- chk7_1     denominators       -- проверка задачи 7-1
- chk7_2     isPrime            -- проверка задачи 7-2

Операторы
- chk8_1     (.+.)              -- проверка задачи 8-1
- chk8_2     (./.)              -- проверка задачи 8-2
- chk8_3     (.-:)              -- проверка задачи 8-3
- chk8_4     (.+:)              -- проверка задачи 8-4
- chk8_5     (.^)               -- проверка задачи 8-5
- chk8_6     (.@)               -- проверка задачи 8-6
- chk8_7     (.*.)              -- проверка задачи 8-7

Списки
- chk9_1     inRange            -- проверка задачи 9-1
- chk9_2     pairs              -- проверка задачи 9-2
- chk9_3     idxs               -- проверка задачи 9-3
- chk9_4     delGt              -- проверка задачи 9-4
- chk9_5     fsearch            -- проверка задачи 9-5
- chk9_6     remdumps           -- проверка задачи 9-6
- chk9_7     chkDups            -- проверка задачи 9-7
- chk9_8     isSorted           -- проверка задачи 9-8
- chk9_9     insWord            -- проверка задачи 9-9
- chk9_10    insWords           -- проверка задачи 9-10
- chk9_11    substrings         -- проверка задачи 9-11
- chk9_12    isPermutation      -- проверка задачи 9-12
- chk9_13    qs                 -- проверка задачи 9-13

Сопоставление с образцом
- chk10_1    pmatch             -- проверка задачи 10-1

Ханойская башня
- chk11_1    hanoiSteps         -- проверка задачи 11-1
- chk11_2    hanoiNum           -- проверка задачи 11-2

Точки на 2D-плоскости
- chk12_1    dist2D             -- проверка задачи 12-1
- chk12_2    dist2Ds            -- проверка задачи 12-2
- chk12_3    minDist2Ds         -- проверка задачи 12-3

Контакты
- chk13_1    emails             -- проверка задачи 13-1
- chk13_2    findContacts       -- проверка задачи 13-2

Операции с полиномами
- chk14_1    eqPP               -- проверка задачи 14-1
- chk14_2    addPP              -- проверка задачи 14-2
- chk14_3    subPP              -- проверка задачи 14-3
- chk14_4    mulPP              -- проверка задачи 14-4
```

```
- chk14_5    divModPP          -- проверка задачи 14-5
- chk14_6    valPX             -- проверка задачи 14-6
- chk14_7    mulCP             -- проверка задачи 14-7
- chk14_8    povPN             -- проверка задачи 14-8
- chk14_9    dPdX              -- проверка задачи 14-9
- chk14_10   dNPdXN            -- проверка задачи 14-10

Операции с полиномами (альтернативное представление)
- chk15_1    sim'P             -- проверка задачи 15-1
- chk15_2    eq'PP             -- проверка задачи 15-2
- chk15_3    add'PP            -- проверка задачи 15-3
- chk15_4    sub'PP            -- проверка задачи 15-4
- chk15_5    mul'PP            -- проверка задачи 15-5
- chk15_6    divMod'PP         -- проверка задачи 15-6
- chk15_7    val'PX            -- проверка задачи 15-7
- chk15_8    mul'CP            -- проверка задачи 15-8
- chk15_9    pov'PN            -- проверка задачи 15-9
- chk15_10   dP'dX             -- проверка задачи 15-10
- chk15_11   dNP'dXN           -- проверка задачи 15-11

Логические формулы
- chk16_1    vars              -- проверка задачи 16-1
- chk16_2    evalPV            -- проверка задачи 16-2
- chk16_3    tauto             -- проверка задачи 16-3
- chk16_4    contra            -- проверка задачи 16-4

Арифметические формулы
- chk17_1    evalEV            -- проверка задачи 17-1
- chk17_2    ddv               -- проверка задачи 17-2

Деревья поиска
- chk18_1    trees             -- проверка задачи 18-1
- chk18_2    nTrees            -- проверка задачи 18-2

Конец списка заданий на автопроверку
```

1 Проба пера

Везде далее приводятся формулировка задач и за ней решение студента. Во всех задачах функции-решения должны быть не определены при некорректном значении аргументов. Конкретная реализация данной неопределенности допускается любой: `undefined`, `error`, недопустимые операции (деление на ноль, `head []` и т. п.), неполные образцы, неполные охранные выражения и т. п. Если в условиях задачи упомянуты несколько параметров задачи, то у функции-решения эти параметры перечисляются в порядке их упоминания в условиях.

1-1 Написать функцию `cube :: Float → Float`, возводящую в куб заданное число.

Мое решение:

```
cube :: Float → Float
cube y = y*y*y
```

1-2 Написать функцию `decmls :: Integer → Integer`, которая вычисляет, сколько существует неотрицательных десятичных чисел с `n` цифрами.

Мое решение:

1-3 Написать функцию `ese :: Integer → Bool`, вычисляющую, будет ли четной сумма `n` четных чисел?

Мое решение:

1-4 Написать функцию `eso :: Integer → Bool`, вычисляющую, будет ли четной сумма `n` нечетных чисел?

Мое решение:

1-5 Написать функцию `epo :: Integer → Bool`, вычисляющую, будет ли четным произведение `n` нечетных чисел?

Мое решение:

1-6 В магазине «Все для чая» есть `n` разных чашек и `m` разных блюдец. Сколькими способами можно купить чашку с блюдцем? Написать функцию, которая это вычисляет:

```
n_cups_m_saucers :: Integer → Integer → Integer
```

Мое решение:

1-7 В Стране Чудес есть три города: *А*, *Б* и *В*. Из города *А* в город *Б* ведет `n` дорог, а из города *Б* в город *В* — `m` дорог. Сколькими способами можно проехать от *А* до *В*? Написать функцию, которая это вычисляет: `wonderland3 :: Integer → Integer → Integer`.

Мое решение:

1-8 В Стране Чудес есть четыре города: *А*, *Б*, *В* и *Г*. Из города *А* в город *Б* ведет `n` дорог, из города *Б* в город *В* — `m` дорог, из города *А* в город *Г* — две дороги и из города *Г* в город *В* — тоже две дороги. Сколькими способами можно проехать от *А* до *В*? Написать функцию, которая это вычисляет: `wonderland4 :: Integer → Integer → Integer`.

Мое решение:

1-9 В магазине «Все для чая» по-прежнему продается `n` чашек, `m` блюдец и 4 чайные ложки. Сколькими способами можно купить два предмета с разными названиями? Написать функцию `fortee :: Integer → Integer → Integer`, которая это вычисляет.

Мое решение:

1-10 Каждую клетку прямоугольной таблицы из `n` строк и `m` столбцов можно покрасить в черный или белый цвет. Сколько существует различных раскрасок этой таблицы? Написать функцию `bwtable :: Integer → Integer → Integer`, которая это вычисляет.

Мое решение:

1-11 Сколькими способами можно заполнить одну карточку в лотерее «Спортпрогноз»? В этой лотерее нужно предсказать итог `n` спортивных матчей. Итог каждого матча — победа одной из команд, либо ничья (счет роли не играет). Написать функцию `lottery :: Integer → Integer`, которая это вычисляет.

Мое решение:

1-12 Имеется три ящика, в каждом из которых лежат шары с номерами от `0` до `n`. Из каждого ящика вынимается по одному шару. Какова вероятность того, что вынуты три единицы? Написать функцию, которая вычисляет эту вероятность: `ones3 :: Integer → Float`.

Эталонное решение и проверочная функция:

1-13 Имеется три ящика, в каждом из которых лежат шары с номерами от `0` до `n`. Из каждого ящика вынимается по одному шару. Какова вероятность того, что вынуты три одинаковых числа? Написать функцию, которая вычисляет эту вероятность: `same3 :: Integer → Float`.

Мое решение:

1-14 Имеется три ящика, в каждом из которых лежат шары с номерами от `0` до `n`. Из каждого ящика вынимается по одному шару. Какова вероятность того, что вынуты три разных числа? Написать функцию, которая это вычисляет: `diff3 :: Integer → Float`.

Мое решение:

1-15 Написать функцию `gcd3 :: Integer → Integer → Integer → Integer`, вычисляющую **НОД**(a, b, c), наибольший общий делитель чисел a, b и c.

Мое решение:

1-16 Написать функцию `lcm2 :: Integer → Integer → Integer`, вычисляющую **НОК**(a, b), наименьшее общее кратное чисел a и b.

Мое решение:

1-17 Написать функцию `lcm3 :: Integer → Integer → Integer → Integer`, вычисляющую **НОК**(a, b, c), наименьшее общее кратное чисел a, b и c.

Мое решение:

1-18 Написать функцию `lcm3gcd :: Integer → Integer → Integer → Integer`, вычисляющую для заданных чисел a, b и c значение выражения **НОК**(**НОД**(a, b), **НОД**(b, c), **НОД**(c, a)).

Мое решение:

1-19 Написать функцию `pps :: [[Integer]] → [Integer]`, объединяющую любое число списков.

Мое решение:

2 Корни уравнений

2-1 Написать функцию `abRoot :: Float → Float → Float`, вычисляющую корень уравнения $ax + b = 0$. Если корней нет или их бесконечно много, то функция не определена.

Мое решение:

2-2 Написать функцию

```
abcRoots :: Float → Float → Float → [Float]
```

которая по заданным коэффициентам a, b и c будет вычислять список корней уравнения $ax^2 + bx + c = 0$. Если корней бесконечно много, то функция не определена.

2-3 Написать функцию `ab_Roots :: Float → Float → [Float]`, которая вычисляет список корней уравнения $ax^2 + bx = 0$. Если корней бесконечно много, то функция не определена.

Мое решение:

2-4 Написать функцию

```
abcdRoots :: Float → Float → Float → Float → [Float]
```

вычисляющую корни уравнения $(ax^2 + bx + c)(x + d) = 0$. Если корней бесконечно много, то функция не определена.

Мое решение:

2-5 Написать функцию

```
abcdeRoots ::
    Float → Float → Float → Float → Float → [Float]
```

вычисляющую корни уравнения $(ax^2 + bx + c)(dx^2 + e) = 0$. Если корней бесконечно много, то функция не определена.

Мое решение:

2-6 Написать функцию `absRoots :: Float → Float → [Float]`, которая вычисляет корни уравнения $|x - a| = b$.

Мое решение:

2-7 Написать функцию

```
abs2Roots :: Float → Float → Float →[Float]
```

вычисляющую корни уравнения $|x-a|+|x-b| = c$. Если корней бесконечно много, то функция не определена.

Мое решение:

2-8 Написать функцию

```
absabs2Roots :: Float → Float → Float →[Float]
```

вычисляющую корни уравнения $||x-a|+|x-b|| = c$. Если корней бесконечно много, то функция не определена.

Мое решение:

2-9 Написать функцию `min_ab_Root :: Float → Float → Float`, которая будет вычислять меньший из корней уравнения $ax^2 + bx = 0$. Если корней нет или их бесконечно много, то функция не определена.

Решение:

2-10 Написать функцию с типом

```
min_abcdRoot :: Float → Float → Float → Float → Float
```

вычисляющую меньший из корней уравнения $(ax^2+bx+c)(x+d) = 0$. Если корней бесконечно много, то функция не определена.

Мое решение:

2-11 Написать функцию

```
min_abcdeRoot ::
    Float → Float → Float → Float → Float → Float
```

вычисляющую меньший из корней уравнения $(ax^2 + bx + c)(dx^2 + e) = 0$. Если корней нет или их бесконечно много, то функция не определена.

Мое решение:

2-12 Написать функцию, которая вычисляет меньший из корней уравнения $|x - a| = b$: `min_absRoot :: Float → Float → Float`. Если корней нет, то функция не определена.

Мое решение:

2-13 Написать функцию

```
min_2absRoot :: Float → Float → Float → Float
```

вычисляющую меньший из корней уравнения $|x - a| + |x - b| = c$. Если корней нет, то функция не определена.

Мое решение:

2-14 Написать функцию

```
min_abs2absRoot :: Float → Float → Float → Float
```

вычисляющую меньший из корней уравнения $||x - a| + |x - b|| = c$. Если корней нет, то функция не определена.

Мое решение:

3 Вычислительные задачи

3-1 Напишите функцию, вычисляющую сумму квадратов всех чисел из отрезка `[1 .. n]`: `sumSq :: Int → Int`.

Мое решение:

3-2 Написать функцию `fibs :: [Integer]`, вычисляющую последовательность чисел Фибоначчи (последовательность начинается с нуля).

Мое решение:

3-3 Написать функцию `bins :: Integer → [Integer]`, вычисляющую список биномиальных коэффициентов для многочлена порядка n.

Мое решение:

3-4 Основание натурального логарифма e можно определить как:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

Напишите функцию, которая вычисляет `eApr` $\epsilon \to e'$ число e с заданной точностью $\epsilon > 0$, суммируя все слагаемые из указанной суммы, большие или равные ϵ: `eApr :: Float → Float`.

Мое решение:

3-5 Для заданного $a \geq 0$ с заданной точностью d требуется вычислить $\sqrt{a}$ при помощи алгоритма Герона, который основан на построении следующей последовательности чисел, сходящейся к значению $\sqrt{a}$:

- $x_1 = 1$;
- $x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$;
- условие достижения требуемой точности $|x_{n+1} - x_n| \leq d$, в этом случае x_{n+1} — результат работы алгоритма Герона.

Написать функцию `sqrtH :: Float → Float → Float`, которая по заданным a и d вычисляет приближение $\sqrt{a}$.

Мое решение:

4 Числа и цифры

Для решения группы задач, связанных с позиционными системами исчисления, рекомендуется (не обязательно, но рекомендуется!) вначале изучить (можно в Википедии) тему позиционных систем (стандартных и симметричных) и написать три вспомогательные функции.

Функция revdigs. Определите функцию перевода в любую позиционную систему (стандартную или симметричную), реверсный порядок цифр, без проверки допустимости аргументов:

```
revdigs :: (Integer → (Integer, Integer)) →
           Integer → [Integer]
```

Первый параметр — функция `dm :: Integer → (Integer, Integer)` отделения младшей цифры от числа. В случае стандартных позиционных систем:

```
dm n = divMod n k
```

где `k` — основание системы.

Мое решение:

Функция revsetun. Определите функцию

`revsetun :: Integer → [Integer]`

перевода в симметричную троичную позиционную систему, реверсный порядок цифр.

Мое решение:

Функция revfigs. Определите функцию

`revfigs :: Integer → Integer → [Integer]`

перевода в любую стандартную k-ичную позиционную систему, с проверкой допустимости аргументов, реверсный порядок цифр.

Мое решение:

4-1 Написать функцию `binary_dgts :: Integer → [Integer]`, переводящую число `n >= 0` в список его двоичных цифр.

Мое решение:

4-2 Написать функцию

`ternary_dgts :: Integer → [Integer]`

переводящую число `n >= 0` в список его троичных цифр.

Мое решение:

4-3 Написать функцию

`decimal_dgts :: Integer → [Integer]`

переводящую число `n >= 0` в список его десятичных цифр.

Мое решение:

4-4 Написать функцию

`k_ary_dgts :: Integer → Integer → [Integer]`

переводящую число `n >= 0` в список его цифр в системе счисления по основанию `k > 1`.

Мое решение:

4-5 Написать функцию

`setun_dgts :: Integer → [Integer]`

переводящую число n в список его цифр в троичной симметричной системе счисления (троичных цифр машины «Сетунь»).

Мое решение:

4-6 Написать функцию `s2dgts :: Integer → Integer`, вычисляющую сумму последних двух цифр десятичной записи числа.

Мое решение:

4-7 Написать функцию `p3dgts :: Integer → Integer`, вычисляющую произведение последних трех цифр десятичной записи числа.

Мое решение:

4-8 Написать функцию `l4dgts :: Integer → [Integer]`, выдающую список из последних четырех цифр числа.

Мое решение:

5 Строки и символы

5-1 Написать функцию `dgt_chr :: Integer → Char`, которая позволяет перевести цифру в символ. Функция не определена, если параметр не из отрезка `[0..9]`.

Мое решение:

5-2 Написать функцию `chr_dgt :: Char → Integer`, которая позволяет перевести символ в цифру. Функция не определена, если параметр не является символом-цифрой.

Мое решение:

5-3 Написать функцию `sumDigits :: String → Int`, складывающую все цифры в строке, например:

```
main:> sumDigits "CA 90210"
12

main:> sumDigits "No digits here!"
0
```

Мое решение:

5-4 Написать функцию `upAlpha :: String → String`, которая выбирает из строки все латинские буквы и переводит их в верхний регистр, например:

```
main:> upAlpha "Hello, World!"
"HELLOWORLD"

main:> upAlpha "Hi."
"HI"
```

Мое решение:

5-5 Пусть имя человека записывается только в одном из трех форматов «фамилия», «имя фамилия», «имя отчество фамилия». Определите функцию `abbrev :: [String] → [String]`, которая в заданном списке имен людей, например:

```
xs = ["Синицин", "Игорь Федорович Поддубный", "Сергей Елкин"]
```

выполняет сокращение имен и отчеств до инициалов:

```
abbrev xs = ["Синицин", "И. Ф. Поддубный", "С. Елкин"]
```

Мое решение:

6 Кредит и его погашение

6-1 Василий Пыжиков берет в кредит `k > 0` долларов. Годовая процентная ставка по кредиту `p > 0`. Ежемесячно Василий обязуется отдавать `x > 0` долларов, и это постепенно гасит кредит. Написать функцию, которая выдает список задолженностей Василия перед банком по месяцам:

`balances :: Float → Float → Float → [Float]`

Первый элемент списка (нулевой месяц, месяц получения кредита) — `k`, последний элемент (месяц завершения выплат по кредиту) — 0. Функция не определена (и это верно для всех функций данного раздела) в случае некорректных исходных данных, в том числе если выплаты малы для погашения кредита.

Мое решение:

6-2 Василий Пыжиков берет в кредит `k > 0` долларов. Годовая процентная ставка по кредиту `p > 0`. Ежемесячно Василий обязуется отдавать `x > 0` долларов, и это постепенно гасит кредит. Написать функцию `balance :: Float → Float → Float → Integer → Float`, которая выдает задолженность Василия перед банком в месяц m.

Мое решение:

6-3 Василий Пыжиков берет в кредит `k > 0` долларов. Годовая процентная ставка по кредиту `p > 0`. Ежемесячно Василий обязуется отдавать `x > 0` долларов, и это постепенно гасит кредит. Через сколько лет он выплатит кредит? Написать функцию `credit_years :: Float → Float → Float → Float`, которая вычисляет эту величину.

Мое решение:

6-4 Василий Пыжиков берет в кредит `k > 0` долларов. Годовая процентная ставка по кредиту `p > 0`. Ежемесячно Василий обязуется отдавать `x > 0` долларов, и это постепенно гасит кредит. Написать функцию, которая выдает список задолженностей Василия перед банком по годам:

`balances_y :: Float → Float → Float → [Float]`

Первый элемент списка — `k` (нулевой год — момент получения кредита), второй элемент — долг перед банком после 1 года (после 12 месяцев), последний элемент — 0 (год, когда завершили выплаты по кредиту).

Мое решение:

7 Делители и простые числа

7-1 Написать функцию `denominators :: Integer → [Integer]`, переводящую число `n /= 0` в список его неотрицательных делителей (по возрастанию).

Мое решение:

7-2 Написать функцию `isPrime :: Integer → Bool` проверки простоты заданного числа $n > 1$.

Мое решение:

8 Операторы

8-1 Написать оператор `(.+.) :: Integer → Integer → Integer`, вычисляющий сумму общих делителей двух чисел.

Мое решение:

8-2 Написать оператор `(./.) :: Integer → Integer → Integer`, который из двух чисел выдает то, у которого больше нулей в десятичной записи.

Мое решение:

8-3 Написать оператор `(.-:) :: Integer → Integer → Integer`, который из двух чисел выдает то, у которого меньше наименьший делитель, больший 1. Если для аргументов (или для одного из них) не удается найти делитель, больший 1, то функция не определена.

Мое решение:

8-4 Написать оператор `(.+:) :: Integer → Integer → Integer`, который из двух чисел выдает то, у которого больше наименьший делитель, больший 1. Если для аргументов (или для одного из них) не удается найти делитель, больший 1, то функция не определена.

Мое решение:

8-5 Определить оператор

```
(.^) :: Float → Integer → Float
```

который в качестве первого параметра принимает начальную цену товара, в качестве второго параметра — количество месяцев и выдает, какая будет цена товара через этот период. Среднегодовой коэффициент инфляции принять равным `0.7`.

Мое решение:

8-6 Написать оператор

```
(.@) :: [String] → String → [String]
```

который в качестве первого параметра принимает список имен (например, «aaron», «asya», «boris»), в качестве второго параметра — имя домена (например, «mail.ru») и выдает список возможных e-mail для рассылки электронных писем («aaron@mail.ru», «asya@mail.ru», «boris@mail.ru»).

Мое решение:

8-7 Написать оператор `(.*.) :: Integer → Integer → Integer`, который из двух чисел выдает то, у которого больше произведение десятичных цифр.

Мое решение:

9 Списки

9-1 Написать функцию `inRange :: Int → Int → [Int] → [Int]`, которая возвращает все элементы списка в заданных рамках (включительно), например:

```
main:> inRange 5 10 [1..15]
[5,6,7,8,9,10]
```

Мое решение:

9-2 Напишите функцию `pairs :: Int → [(Int, Int)]`, которая строит список всех различных пар (x, y), таких, что $1 \leq x \leq n$, $1 \leq y \leq n$.
Мое решение:

9-3 Написать функцию `idxs :: Eq a ⇒ [a] → a → [Int]`, которая находит в списке все вхождения элементов, равных `(==)` заданному, и выдает позиции, в которых он встретился, например:

```
main:> idxs "Bookshop" 'o'
[1,2,6]

main:> idxs "senselessness's" 's'
[0,3,7,8,11,12,14]
```

Мое решение:

9-4 Написать функцию, которая удаляет из списка все элементы, больше заданного: `delGt:: Ord a ⇒ a → [a] → [a]`.
Мое решение:

9-5 Дан список `[Float]` вещественных чисел $x_1 \leq x_2 \leq \ldots \leq x_n$ и дано вещественное число y. Напишите функцию `fsearch :: [Float] → Float → Int`, вычисляющую такое k, что $x_k < y \leq x_{k+1}$.
Особые случаи. Если список пустой или $y \leq x_1$, то функция возвращает `0`. Если $x_n < y$, то функция возвращает n.
Мое решение:

9-6 Написать функцию `remdumps :: Eq a ⇒ [a] → [a]`, которая удаляет копии одинаковых соседних элементов из списка, используя `foldr`:

```
main:> remdumps [1,2,2,3,3,3,1,1]
[1,2,3,1]
```

Мое решение:

9-7 Определите функцию `chkDups :: Eq a ⇒ [a] → Bool`, возвращающую `True`, если в списке, являющемся ее аргументом, дважды содержится хотя бы один элемент.

```
main:> chkDups [1,2,3,4,5]
False

main:> chkDups [1,2,3,2]
True
```

Мое решение:

9-8 Напишите функцию `isSorted :: Ord a ⇒ [a] → Bool`, возвращающую `True`, если ее аргумент — отсортированный список, `True` — иначе.

Мое решение:

9-9 Имеется список слов `ys`, расположенных в алфавитном порядке. Написать функцию `insWord :: String → [String] → [String]` вставки нового слова `x` в список `ys` с сохранением порядка, при условии, что `x` в `ys` нет (если уже есть, то `x` не вставляется в `ys`).

Мое решение:

9-10 Имеется список слов `ys`, расположенных в алфавитном порядке. Написать функцию `insWords :: [String] → [String] → [String]`, которая все слова `x` из `xs` добавляет в `ys`, в соответствии с условиями задачи 9-9.

Мое решение:

9-11 Напишите функцию `substrings :: String → [String]`, которая по заданной строке возвращает все возможные уникальные (неповторяющиеся) подстроки.

Мое решение:

9-12 Перестановкой называется список, состоящий из тех же элементов, что и исходный список, но расположенных в другом порядке. Например, `[1,2,1]` является перестановкой `[2,1,1]`. Напишите функцию

```
isPermutation :: Eq a ⇒ [a] → [a] → Bool
```

возвращающую `True`, если один из ее аргументов является перестановкой другого.

Мое решение:

9-13 Напишите функцию `qs :: Int → [(Int, Int, Int, Int)]`, которая находит список всех различных четверок (a, b, c, d), таких, что числа (a, b, c, d) взаимно простые, $a^2 + b^2 = c^2 + d^2$ и $1 \le a < c \le d < b \le n$.

Мое решение:

10 Сопоставление с образцом

В Unix shell можно писать, например, такие образцы: `*.hs`, которые сопоставляются с некоторым множеством имен файлов.

Формальные правила сопоставления образца p со строкой s следующие:

- пустой образец сопоставляется только с пустой строкой;
- символ звездочка `'*'` сопоставляется с любой (возможно и пустой) последовательностью символов;
- любой другой символ сопоставляется только с таким же символом;

- если образец p' сопоставляется со строкой s', а образец p'' сопоставляется со строкой s'', то образец $p'\,p''$ сопоставляется со строкой $s'\,s''$.

10-1 Напишите функцию `pmatch :: String → String → Bool`, которая проверяет, сопоставляется ли образец `p` со строкой `s`.

Мое решение:

11 Ханойская башня

Головоломка «Ханойская башня» устроена следующим образом. На доске есть три иглы (1, 2, 3). На игле 1 размещена башня из `n` дисков; нижний диск имеет самый большой диаметр, а диаметр каждого следующего диска (над ним) меньше предыдущего.

За один ход с любой иглы $i \in \{1, 2, 3\}$ можно взять один верхний диск и переместить его на другую иглу $j \in \{1, 2, 3\}$. Однако разрешено класть диск лишь либо на доску (то есть до хода игла j была пустой), либо на диск большего диаметра, который до хода был верхним на игле j.

Каждый ход записывают парой `(`i`, `j`) :: (Int, Int)`, а последовательность ходов записывают при помощи списка подобных пар.

Цель головоломки. Нужно найти последовательность ходов для перемещения всей башни с иглы 1 на иглу 3.

Решение головоломки. Рассматривается такой алгоритм решения головоломки. Для переноса башни из n дисков с иглы 1 на иглу 3 (используя вспомогательную иглу 2) необходимо:

- перенести башню из $(n-1)$ дисков с диска 1 на иглу 2, используя вспомогательную иглу 3;
- затем одним шагом перенести нижний диск с иглы 1 на иглу 3;
- затем перенести башню из $(n-1)$ дисков с диска 2 на иглу 3, используя вспомогательную иглу 1.

11-1 Напишите функцию `hanoiSteps :: Int → [(Int, Int)]`, вычисляющую цепочку ходов, необходимых для того, чтобы решить головоломку для заданного `n` — переместить `n` дисков с 1-й иглы на 3-ю.

Мое решение:

11-2 Напишите функцию `hanoiNum :: Integer → Integer`, вычисляющую количество ходов, необходимых для того, чтобы решить головоломку для заданного `n` — переместить `n` дисков с 1-й иглы на 3-ю.

Замечание: функция `hanoiNum` должна вычисляться для весьма больших `n`. Поэтому не следует функцию `hanoiNum` писать с использованием `hanoiSteps` (по идее «построим список ходов, потом посчитаем длину списка»), такой подход не пройдет (например, из-за нехватки памяти).

Мое решение:

12 Точки на 2D-плоскости

О представлении точек на 2D-плоскости читайте в разделе 19.3 на странице 23 (модуль `ExtraTypes`).

12-1 Напишите функцию

```
dist2D :: Point2D → Point2D → Float
```

вычисляющую расстояние $\rho(A(x_1, y_1), B(x_2, y_2))$ между точками.

Мое решение:

12-2 Напишите функцию

```
dist2Ds :: Point2D → [Point2D] → [Float]
```

вычисляющую расстояния от заданной точки до всех точек, перечисленных в списке.

Мое решение:

12-3 Напишите функцию `minDist2Ds :: [Point2D] → Float`, находящую минимальное расстояние между разными точками из заданного списка: `minDist2Ds` $[p_1, \ldots p_n]$ `=` $min\{\rho(p_i, p_j) \,|\, 1 \leq i < j \leq n\}$.

Мое решение:

13 Контакты

О представлении списков контактов читайте в разделе 19.6 на странице 24 (модуль `ExtraTypes`).

13-1 Определите функцию `emails :: Contacts → [Email]`, которая извлекает из списка контактов список (без повторов) всех электронных адресов.

Мое решение:

13-2 Определите функцию

```
findContacts :: Contacts → Name → [(Phone, Email)]
```

которая из списка контактов по имени персоны разыскивает все телефоны и адреса персоны.

Мое решение:

14 Операция с полиномами

О представлении полиномов структурой данных `Poly` читайте в разделе 19.1 на странице 22 (модуль `ExtraTypes`).

14-1 Напишите функцию `eqPP :: Poly → Poly → Bool`, сравнивающую на равенство два многочлена.

Мое решение:

14-2 Напишите функцию `addPP :: Poly → Poly → Poly`, складывающую два многочлена.
Мое решение:

14-3 Напишите функцию `subPP :: Poly → Poly → Poly`, вычитающую из одного полинома другой.
Мое решение:

14-4 Напишите функцию `mulPP :: Poly → Poly → Poly`, перемножающую два полинома.
Мое решение:

14-5 Напишите функцию `divModPP :: Poly → Poly → (Poly, Poly)` деления с остатком одного полинома на другой. Пусть A, B — произвольные полиномы, `(Q, R) = divModPP A B`, пусть (a, b, q, r) — степени полиномов (A, B, Q, R). Тогда должно выполняться $A = Q \times B + R$ и $r < b$.
Мое решение:

14-6 Напишите функцию `valPX :: Poly → Float → Float`, вычисляющую значение `valPX p x` многочлена `p` при значении переменной `x`.
Мое решение:

14-7 Напишите функцию `mulCP :: Float → Poly → Poly`, выполняющую умножение `mulCP c p` многочлена `p` на скаляр `c`.
Мое решение:

14-8 Напишите функцию `povPN :: Poly → Int → Poly`, вычисляющую результат p^n `= povPN p n` возведения полинома p в степень $n \geq 0$.
Мое решение:

14-9 Напишите функцию `dPdX :: Poly → Poly`, выполняющую дифференцирование многочлена.
Мое решение:

14-10 Напишите функцию `dNPdXN :: Poly → Int → Poly`, вычисляющую производную `dNPdXN p` n заданного порядка $n \geq 0$ от полинома `p`.
Мое решение:

15 Операции с полиномами с альтернативным представлением

Об альтернативном представлении полиномов структурой данных `Poly` читайте в разделе 19.2 на странице 23 (модуль `ExtraTypes`).

При альтернативном представлении полинома полезно работать с «упрощенным» полиномом. В упрощенном полиноме приведены подобные члены, удалены члены с нулевым коэффициентом, все мономы отсортированы: левее — мономы с младшими степенями, правее — мономы со старшими степенями.

15-1 Определите функцию `sim'P :: Poly' → Poly'`, упрощающую полином (см. пояснения выше).

Мое решение:

15-2 Напишите функцию `eq'PP :: Poly' → Poly' → Bool`, сравнивающую на равенство два многочлена.

Мое решение:

15-3 Напишите функцию `add'PP :: Poly' → Poly' → Poly'`, складывающую два многочлена. Результат функция `add'PP` должна возвращать в упрощенном виде.

Мое решение:

15-4 Напишите функцию `sub'PP :: Poly' → Poly' → Poly'`, вычитающую из первого аргумента (полинома), второй (полином). Результат функция `sub'PP` должна возвращать в упрощенном виде.

Мое решение:

15-5 Напишите функцию `mul'PP :: Poly' → Poly' → Poly'`, перемножающую два полинома. Результат функция `mul'PP` должна возвращать в упрощенном виде.

Мое решение:

15-6 Напишите функцию

```
divMod'PP :: Poly' → Poly' → (Poly', Poly')
```

деления с остатком одного полинома на другой.

Мое решение:

15-7 Напишите функцию `val'PX :: Poly' → Float → Float`, вычисляющую значение многочлена `p` для заданного значения `x` переменной.

Мое решение:

15-8 Напишите функцию `mul'CP :: Float → Poly' → Poly'`, выполняющую умножение многочлена на скаляр.

Мое решение:

15-9 Напишите функцию `pov'PN :: Poly' → Int → Poly'`, вычисляющую результат p^n = `pov'PN` p n возведения полинома p в степень $n \geq 0$.

Мое решение:

15-10 Напишите функцию, вычисляющую первую производную многочлена: `dP'dX :: Poly' → Poly'`.

Мое решение:

15-11 Напишите функцию `dNP'dXN :: Poly' → Int → Poly'`, вычисляющую производную заданного порядка от полинома.

Мое решение:

16 Логические формулы

О представлении логических формул читайте в разделе 19.4 на странице 23 (модуль `ExtraTypes`).

16-1 Напишите функцию `vars :: Prop → [String]`, которая возвращает список всех переменных (без повторов), использованных в логической формуле.

Мое решение:

16-2 Пусть заданы логическая формула и означивание всех переменных из нее некими логическими значениями. Означивание задается в виде списка пар «переменная — значение», например: `[("p", True), ("q", False)]`. Напишите функцию `evalPV :: Prop → [(String,Bool)] → Bool`, которая определяет, истинна ли логическая формула при заданных значениях переменных.

Мое решение:

16-3 Напишите функцию `tauto :: Prop → Bool`, которая возвращает значение `True`, если логическая формула истинна при любых значениях переменных, и значение `False` в остальных случаях.

Мое решение:

16-4 Напишите функцию `contra :: Prop → Bool`, которая возвращает значение `True`, если логическая формула ложна при любых значениях переменных, и значение `False` в остальных случаях.

Мое решение:

17 Арифметические выражения

О представлении арифметических выражений читайте в разделе 19.5 на странице 24 (модуль `ExtraTypes`).

17-1 Пусть задано арифметическое выражение и означивание всех переменных из выражения некими значениями. Означивание задается в виде списка пар «переменная — значение», например: `[("p", -1.7), ("q", 3.14)] :: [(String,Float)]`. Напишите функцию, вычисляющую значение арифметического выражения при заданных значениях переменных:

```
evalEV :: Expr → [(String,Float)] → Float
```

Мое решение:

17-2 Напишите функцию `ddv :: Expr → String → Expr` дифференцирования выражения, принимающую на вход выражение и переменную, по которой происходит дифференцирование.

Мое решение:

18 Деревья поиска

О представлении деревьев поиска читайте в разделе 19.7 на странице 24 (модуль `ExtraTypes`).

18-1 Пусть `a` тип такой, что `Ord a`, `xs :: [a]` — отсортированный список конечной длины без повторов элементов. Определите функцию

```
trees :: Ord a ⇒ [a] → [Tree a]
```

которая по списку `xs` строит список всех различных возможных деревьев `t`, таких, что `xs == flatten t`.

Мое решение:

18-2 Определите функцию `nTrees :: Int → Integer`, которая по заданному числу `n` вычисляет число всех различных возможных деревьев `t`, таких, что `[1..n] == flatten t`.

Замечание: функция `nTrees` должна вычисляться для весьма больших `n`. Поэтому не следует функцию `nTrees` писать с использованием `trees` (например, по схеме `length (trees [1..n])`), такой подход не пройдет (из-за огромных необходимых ресурсов: времени счета и памяти).

Мое решение:

19 Модуль ExtraTypes: типы данных, определенные для решения некоторых задач

В модуле:

```
module ExtraTypes where
```

предопределены типы данных, необходимые для решения задач (рассмотрены далее).

19.1 Полиномы от одной переменной

Многочлен степени n от переменной x может быть представлен списком его коэффициентов:

```
newtype Poly = MkPoly [Float]
               deriving Show
```

Коэффициенты в списке следует размещать в порядке возрастания степеней x. Например, многочлен:

$$3.1x^4 + 4.2x^3 + 9.3x + 7.4,$$

который есть в точности многочлен:

$$7.4 + 9.3x + 0.0x^2 + 4.2x^3 + 3.1x^4,$$

представляется списком:

`MkPoly [7.4, 9.3, 0.0, 4.2, 3.1]`.

19.2 Альтернативное представление полинома от одной переменной — «разряженный список»

Кроме рассмотренного ранее представления полиномов от одной переменной x (см. раздел 19.1), возможно представление таких полиномов еще в виде «разряженного списка». При подобном подходе полином:

$$3.4x^3 + 2x^4 + 1.5x^3 + 7.1x^5$$

будет представлен следующим образом:

```
[(3.4, 3), (2.0, 4), (1.5, 3), (7.1, 5)] :: [(Float, Int)]
```

Определим тип-синоним для представления полинома в виде «разряженного списка»:

```
type Poly' = [(Float, Int)]
```

19.3 Точки на двумерной плоскости

Точка на двумерной плоскости может быть представлена парой ее координат:

```
newtype Point2D = MkP2D (Float, Float)
                  deriving Show
```

19.4 Логические формулы

Логической формулой p называют формулу вида:

- имя переменной — строка;
- булевская константа — `True` или `False`;
- $p' \wedge p''$;
- $p' \vee p''$;
- $\neg p'$

где p' и p'' — логические формулы.

Определим тип данных `Prop` для представления логических формул:

```
data Prop = Var String
          | Const Bool
          | And Prop Prop
          | Or  Prop Prop
          | Not Prop
          deriving Show
```

19.5 Арифметическое выражение

Определим арифметическое выражение следующим образом:

- число `n`;
- переменная `x`, где `x :: String`;
- сумма двух выражений;
- произведение двух выражений.

Определим тип данных «арифметическое выражение» `Expr`:

```
data Expr = VarF String
          | ConstF Float
          | Add Expr Expr
          | Mul Expr Expr
          deriving Show
```

19.6 Список контактов

Структуры данных для хранения списка контактов зададим типами-синонимами:

```
type Contact  = (Name, Phone, Email)
type Name     = String
type Phone    = String
type Email    = String
type Contacts = [Contact]
```

19.7 Дерево поиска

Дерево поиска будет представлено ровно так, как это обсуждалось на лекциях:

```
data Tree a = Node a (Tree a) (Tree a) | Leaf
     deriving (Eq, Ord, Show)
```