Лабораторная работа №3
«Бесконечные вычисления»

Предварительные сведения

В ходе выполнения данной работы необходимо написать решение 25 задач на языке Haskell.

Прежде чем приступать к выполнению заданий, ознакомьтесь с замечаниями изложенными в пункте 2 данного описания.

1. Задания

1. Опишите значение `nat` — бесконечный список всех натуральных чисел (целых чисел от 1 до бесконечности).

   Типовое решение — 1 строка кода.

2. Опишите значение `fibonacci` — бесконечную последовательность Фибоначчи, состоящую из целых чисел, первые два числа в которой — 0 и 1. Последовательность Фибоначчи — последовательность, в которой каждый элемент (за исключением первых двух) равен сумме двух предыдущих.

   Решение — одна строка.

3. Опишите значение `factorial` — бесконечную последовательность целых чисел, в которой на n-й позиции стоит $n!$ — факториал числа n. Факториал числа n определяется как произведение всех чисел от 1 до n, причем $0! = 1$. Считаем, что позиции списка нумеруются с нуля.

   Решение — одна строка.

4. Опишите функцию `powerSeq x`, где `x` — число, возвращающую бесконечную последовательность, в которой на i-й позиции стоит i-я степень числа `x`. Считаем, что позиции списка нумеруются с нуля.

   Решение — одна строка.

5. Опишите функцию `findCloseEnough eps stream`, где `eps` — число и `stream` — бесконечный список чисел $\{a_i\}_{i=0}^{\infty}$. Функция должна вернуть такой элемент a_n заданного списка, для которого разность по модулю с предыдущим элементом $|a_n - a_{n-1}|$ не превышает `eps`.

   Типовое решение состоит из пяти строк.

6. Опишите функцию `streamSum stream`, где `stream` — бесконечная последовательность чисел. Функция должна формировать бесконечную последовательность частичных сумм элементов заданной последовательности, т. е. на i-й позиции результата должна стоять сумма первых i элементов списка `stream`. Считаем, что позиции последовательности нумеруются с нуля.

   Решение — 1 строка

7. Опишите функцию `expSummands`, получающую в качестве аргумента вещественное число . Функция должна возвращать последовательность, в которой на i-й позиции стоит значение $x^i/i!$. Считаем, что позиции последовательности нумеруются с нуля.

   Для преобразования целого числа в вещественное потребуется функция `fromIntegral`.

   Объем решения — две строки.

8. Опишите функцию `expStream`, получающую в качестве аргумента вещественное число x и возвращающую последовательность приближений к значению e^x. Считаем, что позиции последовательности нумеруются с нуля.

   Напомним, что

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{2 \cdot 3} + \frac{x^4}{2 \cdot 3 \cdot 4} + \cdots + \frac{x^n}{n!} + \cdots$$

i-м приближением бесконечного ряда будем считать сумму первых i слагаемых этого ряда.

Типовое решение — 1 строка.

9. Опишите функцию `expAppr eps`, где `eps` — вещественное число. Функция должна выдавать функцию, приближенную к e^x с точностью `eps`.

   Функция должна извлекать из последовательности приближений к e^x значение, разность по модулю которого с предыдущим элементом последовательности не превышает `eps`.

   Решение — 1 строка.

10. Производной вещественнозначной функции f называется функция

$$f'(x) = \lim_{dx \to 0} \frac{f(x+dx) - f(x)}{dx}.$$

    Опишите функцию `derivativeAppr f dx`, где `f` — вещественнозначная функция одного аргумента, а `dx` — некоторая малая величина (вещественного типа). Результатом вызова `derivativeAppr f dx` должна быть функция, вычисляющая приближенное значение производной функции `f` для заданного приращения `dx` (т. е. вычисляющую значение выражения для производной без вычисления предела).

    Объем решения составит 1-2 строки.

11. Опишите функцию `derivativeStream f`, где `f` — вещественнозначная функция одного аргумента, выдающую бесконечную последовательность приближений к производной функции `f` для приращений из последовательности

$$1,\ \frac{1}{2},\ \frac{1}{4},\ \frac{1}{8},\ \ldots,\ \frac{1}{2^n},\ \ldots$$

    Объем решения — две строки.

12. Опишите функцию `derivative f`, выдающую функцию — производную функции `f`. В качестве решения функция должна выдавать элемент последовательности `derivativeStream f`, для которого разность по модулю с предыдущим элементом даст значение, не превышающее `epsilon'` (константа, заданная в шаблоне с решением).

    Типовое решение — 1 строка.

13. Вычисление значения обратной функции для функции f в точке x с начальным приближением y_0 можно представить в виде разложения в степенной ряд:

$$f^{-1}(x) = \sum_{k=0}^{\infty} A_k(f', y_0) \frac{(x - f(y_0))^k}{k!}, \quad \text{где } A_0(g, x) = x,\ A_k(g, x) = \frac{A'_{k-1}(x)}{g(x)}$$

    Опишите функцию `funAkStream g`, формирующую для заданной функции `g` бесконечную последовательность функций от x, выдающих $A_k(g, x)$.

    Опишите функцию `invF f y0`, где `f` — вещественнозначная функция, `y0` — вещественное число — начальное приближение. Результатом `invF` должна быть функция — приближение к обратной функции для `f`.

    Для этого в теле функции `invF` сначала сформируйте последовательность частичных сумм ряда, из которой с помощью `findCloseEnough` найдите решение с точностью `epsilon` (константа, заданная в шаблоне с решением).

    ВНИМАНИЕ! При тестировании функции `invF` значение `y0` должно задаваться достаточно близким к ожидаемому результату. В противном случае выполнение функции может быть слишком долгим/бесконечным.

    Объем типового решения — 2 строки для `funAkStream` и 7 строк для `invF`.

14. Опишите функцию `average x y`, находящую среднее арифметическое числовых значений `x` и `y`.

    Решение — 1 строка.

15. Опишите функцию `averageDump f`, где `f` — вещественнозначная функция. Результатом должна быть функция, выдающая для каждого значения x среднее арифметическое между x и результатом вычисления функции `f` от этого значения.

    Решение — 1 строка.

16. Опишите функцию `newtonTransform g`, где `g` — вещественнозначная функция. Результатом должна быть функция, которая для каждого значения x вычисляет значение

$$x - \frac{\mathrm{g}(x)}{\mathrm{g}'(x)}.$$

Решение — 1 строка.

17. Некоторые сходящиеся последовательности можно «ускорить»: получить по специальному закону новую последовательность, сходящуюся к тому же пределу, но с большей скоростью. Одним из таких ускорителей является формула Эйткена

$$\sigma_n = \frac{s_{n+1}s_{n-1} - s_n^2}{s_{n+1} - 2s_n + s_{n-1}},$$

по которой для каждых трех соседних элементов исходной последовательности $\{s_i\}_{i=0}^{\infty}$ получается один элемент последовательности $\{\sigma_i\}_{i=1}^{\infty}$.

Опишите функцию `eitken`, получающую в качестве аргумента последовательность и выдающую новую последовательность, полученную по формуле Эйткена.

Решение — 2 строки.

18. Неподвижной точкой функции f является такое значение x^*, что $f(x^*) = x^*$. Для некоторых функций f можно найти неподвижную точку, начав с какого-то значения x_0 и применяя f многократно:

$$f(x),\ f(f(x)),\ f(f(f(x))),\ \ldots$$

— пока значение не перестанет сильно изменяться.

Опишите функцию `fixedPoint f x0`, где `f` — вещественнозначная функция, `x0` — начальное приближение. Используя идею, изложенную выше, функция должна выдавать бесконечную последовательность приближений к неподвижной точке функции `f`, начинающуюся с `x0`.

Решение — 1 строка.

19. Опишите функцию `fixedPointOfTransform f transform x0`, где `f` — вещественнозначная функция, `x0` — начальное приближение, `transform` — функция для преобразования вещественнозначной функции (функция, получающая функцию и выдающая функцию).

`fixedPointOfTransform` должна вычислять с точностью `epsilon'` предел последовательности приближений к неподвижной точке с начальным приближением `x0` результата преобразования `transform` функции `f`.

Типовое решение — 2 строки.

20. Рассмотрим функцию извлечения квадратного корня $y = \sqrt{x}$. Из равенства следует, что $y^2 = x$. Последнее равенство при ненулевом значении y можно представить в виде $y = \frac{x}{y}$. Отсюда можно сделать вывод, что для фиксированного x квадратный корень из x есть значение неподвижной точки функции

$$f(y) = \frac{x}{y}.$$

Но для такой функции алгоритм поиска неподвижной точки, реализованный функцией `fixedPoint`, неприменим (проверьте это). Вместо этого можно воспользоваться следующим свойством: если y^* является неподвижной точкой функции f, то y^* является также неподвижной точкой функции

$$g(y) = \frac{y + f(y)}{2}.$$

Функцию g для заданной функции f вам выдаст преобразование `averageDump`.

Воспользуйтесь изложенными идеями для описания функции `sqrt1 x`, выдающей значение квадратного корня из `x`. В качестве начального приближения при поиске неподвижной точки возьмите 1.0.

Решение — 1 строка.

21. Используя идеи, изложенные в предыдущем задании, опишите функцию `cubert1 x` извлечения корня третьей степени из `x`.

Типовое решение — 1 строка.

22. Известно следующее утверждение. Если $x = g(x)$ есть дифференцируемая функция, то решение уравнения $g(x) = 0$ есть неподвижная точка функции $x = f(x)$, где

$$f(x) = x - \frac{g(x)}{g'(x)}. \tag{1}$$

У нас уже есть функция `newtonTransform`, возвращающая такую функцию f для заданной функции g.

Опять рассмотрим задачу извлечения квадратного корня. Снова $y = \sqrt{x}$ преобразуем в $y^2 = x$. Отсюда получаем уравнение для фиксированного x: $y^2 - x = 0$. То есть получили уравнение $g(y) = 0$, где $g(y) = y^2 - x$. Теперь для поиска y можем применить вышеупомянутое утверждение.

Воспользуйтесь изложенными идеями для описания функции `sqrt2 x`, выдающую значение квадратного корня из `x`. В качестве начального приближения при поиске неподвижной точки возьмите 1.0.

Решение — 1 строка.

23. Используя идеи, изложенные в предыдущем задании, опишите функцию `cubert2 x` извлечения корня третьей степени из `x`.

Типовое решение — 1 строка.

24. Опишите функцию `extremum f`, где `f` — вещественнозначная функция. Результатом вызова `extremum f` должна быть пара `(Double,[Char])`, в которой первый элемент — точка x одного из экстремумов функции `f`, а второй — слово `"minimum"`, если в x достигается локальный минимум функции `f`, `"maximum"`, если в x достигается локальный максимум функции `f`, или `"inflection"`, если x — точка перегиба (или если невозможно сделать вывод из знаний первой и второй производных).

Функция должна находить первую производную, приравнивать ее к нулю и находить точку, подозрительную на экстремум — это первый элемент пары-результата. Второй элемент пары должен быть получен на основании анализа знака второй производной: если вторая производная больше нуля, то имеем точку минимума, если меньше нуля — максимума и если равна нулю, то имеем точку перегиба (или ничего сказать не можем).

Вместо равенства нулю следует оценивать попадание точки в интервал `[-epsilon, epsilon]`.

Типовое решение — 9 строк.

25. Опишите получение значения числа π в виде вычисления `myPi`. Вычисление должно быть организовано на основе получения суммы ряда:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots + \frac{(-1)^n}{2n+1} + \cdots.$$

Нужно сформировать последовательность слагаемых ряда, сформировать последовательность частичных сумм, ускорить последовательность частичных сумм с помощью формулы Эйткена и найти предел полученной последовательности с точностью `epsilon''`, а получившийся результат домножить на 4.

Форматированное решение — 5 строк.

Общий объем решения, включая строки, предварительно заданные в файле, должен составить порядка 150 строк.

2. Замечания по выполнению заданий

2.1. Необходимый минимум

Для выполнения работы потребуются сведения о следующих функциях, операциях и конструкциях:

- конструкция `=` для определения функций и переменных
- конструкция `if...then...else...`
- конструкция `let...in...`
- конструкция `...where...`
- конструкция `\x -> ...` для определения неименованной функции
- арифметические операции `+`, `-`, `*`, `/`
- логические операции `||`, `&&`, `not`
- операции сравнения `<`, `>`, `==`, `/=`, `<=`, `>=`
- конструктор кортежа `(,)`

- конструкторы списка `:` и `[]`
- функции работы со списками `head`, `tail`, `map`, `zipWith`, `find`
- функция `iterate` для получения бесконечной последовательности
- функция `abs` нахождения модуля числа
- функция преобразование целого числа в вещественное `fromIntegral`
- конструкторы значений типа `Maybe` — `Just` и `Nothing`
- конструкция `:load` для загрузки функций из файла с заданным именем
- конструкция `:type` для определения типа значения

Информацию об этих и других конструкциях можно найти в конспекте, приведенном на портале.

2.2. Ограничения

При выполнении данной лабораторной работы нужно соблюдать следующие ограничения:

1. При описании функций не должны быть явно описаны типы аргументов и тип результата;
2. Нельзя использовать функции или конструкции, не перечисленные в разделе 2.1. Если вы считаете, что для выполнения какого-то из заданий необходима функция/конструкция, отсутствующая в разделе 2.1, — задайте вопрос на форуме «Лабораторная работа №3»;
3. Нигде, за исключением задания 1, нельзя использовать генераторы списков. В тех заданиях, результатом которых должна быть бесконечная последовательность, результат должен получаться как результат выполнения операций над другими последовательностями или как результат функции `iterate`.

Можно определять собственные вспомогательные функции. Но вспомогательная функция может быть определена как функция верхнего уровня только если она используется в решениях минимум двух разных заданий. В остальных случаях вспомогательные функции должны быть локальными.

2.3. Предостережения насчет языка

Будьте внимательны:

- строковые литералы заключаются в двойные кавычки, а не в апострофы;
- в каждой конструкции `if` блок `else` является неотъемлемой частью;
- унарный минус обозначается обычным образом, как минус `-`;
- сравнение на равенство — `==`, а не `=`;
- сравнение на неравенство — `/=`;
- вместо `and` и `or` здесь `&&` и `||`, но отрицание здесь — `not`;
- есть неявное преобразование типов, но иногда необходимо применить явное (например, при делении на целое число нужно целое превратить в вещественное).

2.4. Предостережения насчет решения

Решением каждой задачи должна быть функция с указанным именем и возвращающая значение в той форме, в которой спрашивается в задании. Прежде чем отправить решение на проверку проводите сравнение сигнатуры написанной вами функции с соответствующей сигнатурой, приведенной в разделе 2.5. В частности, обратите внимание, что функции многих аргументов задаются в каррированной форме. Вспомогательные функции могут определяться в любой форме.

Не следует делать предположений насчет задания, не сформулированных явно в условии. Если возникают сомнения — задайте вопрос на форуме «Лабораторная работа №3».

Избегайте повторений вычислений. Вместо того, чтобы вычислять одно и то же значение несколько раз — сохраняйте вычисленное значение в переменной.

2.5. Результат

Запуск корректно выполненного решения должен привести к следующим определениям:

```
nat :: [Integer]
fibonacci :: [Integer]
factorial :: [Integer]
powerSeq :: Num a => a -> [a]
findCloseEnough :: (Num a, Ord a) => a -> [a] -> a
streamSum :: Num a => [a] -> [a]
expSummands :: Fractional a => a -> [a]
expStream :: Double -> [Double]
expAppr :: Double -> Double -> Double
derivativeAppr :: Fractional a => (a -> a) -> a -> a -> a
```

```haskell
derivativeStream :: Fractional a => (a -> a) -> a -> [a]
derivative :: (Double -> Double) -> Double -> Double
funAkStream :: (Double -> Double) -> [Double -> Double]
invF :: (Double -> Double) -> Double -> Double -> Double
average :: Fractional a => a -> a -> a
averageDump :: Fractional a => (a -> a) -> a -> a
newtonTransform :: (Double -> Double) -> Double -> Double
eitken :: Fractional a => [a] -> [a]
fixedPoint :: (a -> a) -> a -> [a]
fixedPointOfTransform :: a -> (a -> Double -> Double) -> Double -> Double
sqrt1 :: Double -> Double
cubert1 :: Double -> Double
sqrt2 :: Double -> Double
cubert2 :: Double -> Double
extremum :: (Double -> Double) -> (Double,[Char])
myPi :: Double
```