**Вариант 1. Электронный каталог.**

Задан текстовый файл со списком имен и адресов электронной почты. Например:

Николай Андреевич Римский-Корсаков rimsky@mail.ru

Родион Щедрин schedrin@yandex.ru

Сергей Прокофьев prokofiev@rambler.ru

Рейнгольд Морицевич Глиэр glier@mail.ru

Формат: имя отчество (может отсутствовать) фамилия адрес. Также задан текстовый файл со списком адресов электронной почты и номеров ICQ. Например:

balakirev@gmail.com 112114

glier@mail.ru 2356154

prokofiev@rambler.ru 1234321

schedrin@yandex.ru 2232246

Во втором файле могут быть не все адреса из первого файла, могут быть лишние адреса, и порядок может быть другим. Между словами может быть произвольное число пробелов и символов табуляции. Создайте программу, которая на основе этих файлов создает web-страницу, которая в браузере выглядит следующим образом (список должен быть упорядочен по фамилиям, щелчок на ссылке должен отсылать письмо на выделенный адрес):

* Глиэр Рейнгольд Морицевич, [glier@mail.ru](mailto:glier@mail.ru), ICQ: 2356154.
* Прокофьев Сергей, [prokofiev@rambler.ru](mailto:prokofiev@rambler.ru), ICQ: 1234321.
* Римский-Корсаков Николай Андреевич, [rimsky@mail.ru](mailto:rimsky@mail.ru)
* Щедрин Родион, [schedrin@yandex.ru](mailto:schedrin@yandex.ru), ICQ: 2232246.

 HTML-код данной страницы выглядит следующим образом:

<ul>

<li>Глиэр Рейнгольд Морицевич, <a href="mailto:glier@mail.ru">glier@mail.ru</a>, ICQ: 2356154.</li>

<li>Прокофьев Сергей, <a href="mailto:prokofiev@rambler.ru">prokofiev@rambler.ru</a>, ICQ: 1234321.</li>

<li>Римский-Корсаков Николай Андреевич, <a href="mailto:rimsky@mail.ru">rimsky@mail.ru</a></li>

<li>Щедрин Родион, <a href="mailto:schedrin@yandex.ru">schedrin@yandex.ru</a>, ICQ: 2232246.</li>

</ul>

Для определения адреса электронной почты можно использовать наличие в слове символа @. Для этого определите функцию isEmail :: String -> Bool, которая возвращает True, если строка является адресом.

Определите функцию nameList :: String -> [(String, String)], которая преобразует входной файл в список пар имен и адресов (можно использовать функции lines, words, unwords):

[("Николай Андреевич Римский-Корсаков","rimsky@mail.ru"),

("Родион Щедрин","schedrin@yandex.ru"),

("Сергей Прокофьев","prokofiev@rambler.ru"),

("Рейнгольд Морицевич Глиэр","glier@mail.ru")]

В полученном списке перенесите последнее слово первого элемента пары на первое место, и отсортируйте полученный список функцией sort.

Определите функцию formatNames :: [(String, String)] -> String, которая преобразует список, полученный на предыдущем этапе, в строку с кодом HTML.

Определите generateHTML :: IO(), которая преобразует файл, заданный в командной строке первым параметром, в файл с таким же именем и расширением .htm, включая в него номера ICQ из файла, заданного вторым параметром.

**Вариант 2. Экспертная система.**

Создайте простую экспертную систему, которая классифицирует что-либо в зависимости от ответов пользователя на вопросы. Примерами подобных систем могут быть программа поиска причин неисправности или классификации объектов по категориям.

Данная экспертная система будет определять животное из трех возможных: муравей, зебра и золотая рыбка. Система будет использовать таблицу, содержащую вопросы и действия, которые необходимо предпринять при ответах да и нет. Пример: Это живет в воде? Если да, то золотая рыбка. Если нет, то это черное? Если да, то муравей, если нет, то зебра.

Такая таблица может быть представлена типом data Table = Decision String | Question String Table Table, где Decision d означает выдать решение d, а Question q y n означает задать вопрос q, и следовать инструкциям в таблице y или n в зависимости от ответа.

Определите функцию decide :: Table -> IO (), которая следуя инструкциям из таблицы, задает пользователю вопросы и в зависимости от его ответа задает дальнейшие вопросы или выводит результат.

Для создания таблицы нам необходимы вопросы о каждом из животных и ответы на них. Их можно представить в виде списка, где каждое решение связано с вопросом и соответствующим ему ответом. Например:

facts :: Facts

facts =

[("Это муравей",

[("Это меньше карандаша", True),

("Это черное", True),

("Это живет в воде", False)]),

("Это зебра",

[("Это меньше карандаша", False),

("Это живет в воде", False),

("Это полосатое", True),

("Это черное", False)]),

("Это золотая рыбка",

[("Это меньше карандаша", True),

("Это полосатое", True),

("Это черное", False),

("Это живет в воде", True)])]

Определите функцию restrict :: String -> Bool -> Facts -> Facts, которая по вопросу, ответу и списку возможных фактов возвращает список тех фактов, которые данный ответ НЕ исключает. Заметьте, что некоторые факты могут не относиться к вопросу, поэтому они не исключаются любым ответом. Примеры:

restrict "Это полосатое" False facts ==

[("Это муравей",

[("Это меньше карандаша", True),

("Это черное", True),

("Это живет в воде", False)])]

restrict "Это черное" True facts ==

[("Это муравей",

[("Это меньше карандаша", True),

("Это живет в воде", False)])]

Определите функцию table :: Facts -> Table, которая строит таблицу решений по набору фактов. Необходимо обеспечить получение решения за минимальное количество вопросов. В примере для идентификации животного требуется максимум 2 вопроса, хотя по каждому животному известно три или четыре факта. Хорошей стратегией для этого является задание вопроса, который при любом ответе минимизирует количество возможных решений. В данном случае первый вопрос «Это живет в воде» гарантирует, что независимо от ответа остается максимум две возможности. Напротив, вопрос «Это полосатое» при ответе да не исключает ни одного животного (муравьи не полосатые, но этого факта в базе нет).

Напишите программу, реализующую данную функциональность. Предусмотрите возможность загрузки фактов из текстовых файлов.

**Вариант 3. Анализ социологических опросов.**

Разработайте программу для анализа и обобщения результатов опроса. Пример опроса:

1. Каково Ваше общее мнение об этом курсе?
2. Ваше мнение о лекциях?
3. Что Вы думаете о том, что занятия назначены на субботу?
4. Что Вы думаете о заданиях для лабораторных работ?
5. Насколько трудным был для Вас этот курс?

Каждый вопрос имеет фиксированное количество ответов (например: очень хорошо, хорошо, нормально, плохо, очень плохо). Ответ одного человека представлен списком пар (номер вопроса, ответ), например: [(1, "очень хорошо"), (3, "плохо"), (4, "нормально")]. Допускается, что человек мог не ответить на некоторые вопросы (в нашем примере это 2 и 5), но не допускается несколько ответов на один и тот же вопрос. Определите функцию validReply :: Reply -> Bool, которая определяет, допустим ли ответ.

Пример списка ответов:

someReplies :: [Reply]

someReplies =

[[(1, "очень хорошо"), (3, "плохо"), (4, "нормально")],

[(2,"хорошо"), (3,"плохо"), (4,"хорошо"), (5,"трудно")],

[(4,"нормально"), (5,"очень трудно")]]

Определите функцию questions :: [Reply] -> [Int], которая по списку результатов опроса возвращает упорядоченный список номеров вопросов, которые были отвечены хоть в каких-то результатах. Пример: questions someReplies возвращает [1,2,3,4,5].

Определите функцию answers :: Int -> [Reply] -> [String], которая по номеру вопроса и списку результатов опроса возвращает список всех ответов на него. Пример: answers 3 someRepliesвозвращает ["плохо","плохо"].

Определите функцию summary :: [Reply] -> [(Int,[(Int,String)])], которая по списку результатов опроса возвращает таблицу, содержащую для каждого вопроса все ответы, данные на него и количество каждого ответа. Ответы должны быть отсортированы так, чтобы самые частые были вначале. Пример:

Main> summary someReplies

[(1,[(1,"очень хорошо")]), (2,[(1,"хорошо")]), (3,[(2,"плохо")]), (4,[(2,"нормально"),(1,"хорошо")]), (5,[(1,"очень трудно"),(1,"трудно")])]

Определите summarize :: [Reply] -> IO (), которая по списку результатов опроса печатает таблицу ответов с процентами. Пример:

Main> summarize someReplies

Вопрос 1: 100% очень хорошо

Вопрос 2: 100% хорошо

Вопрос 3: 100% плохо

Вопрос 4: 67% нормально 33% хорошо

Вопрос 5: 50% очень трудно 50% трудно

Для более обобщенного анализа сгруппируем ответы со словом «очень» и без него. Определите функцию mild :: [Reply] -> [Reply], которая по списку результатов опроса выдает список результатов опроса без слова «очень». Пример:

Main> summarize (mild someReplies)

Вопрос 1: 100% хорошо

Вопрос 2: 100% хорошо

Вопрос 3: 100% плохо

Вопрос 4: 67% нормально 33% хорошо

Вопрос 5: 100% трудно

Напишите программу, которая получала бы данные из файла, где в строке записаны ответы на анкету одного человека, и выводила бы обобщенные результаты в текстовом виде (см. выше) и в виде столбчатой диаграммы.

**Вариант 4. Система управления расписаниями.**

Создайте систему управления расписаниями сотрудников, которая будет хранить периоды времени, в течение которых сотрудник занят, и предлагать время совещания группы людей, когда все они свободны.

Определите тип Time, который бы хранил запись о времени, определите для него методы show и read, для преобразования к виду, например, 08:15. Время можно хранить как целое число минут, прошедших с 8:00 текущего дня или в любой другой удобной форме.

Временной интервал будем хранить в виде type Period = (Time, Time). Время окончания должно быть строго больше времени начала. При создании периода необходимо проверять это свойство.

Определите функцию before :: Period -> Period -> Bool, которая определяет, находится ли первый период строго раньше второго.

Определите функцию overlap :: Period -> Period -> Bool, которая определяет, пересекаются ли периоды.

Список периодов, в течение которых человек занят, можно хранить в виде type Diary = [Period], при этом каждый период в списке должен быть строго до следующих за ним. Определите функции добавления нового периода на нужное место в список и удаления периода из списка.

Определите функцию findTime :: Int -> Time -> Diary -> Period, которая возвращает по списку Diary самый ранний свободный период длиной Int минут начиная со времени Time, который не пересекается с другими периодами из списка. Например: findTime 5 0 [(0, 10), (15, 20), (30, 40)] == (21, 25).

Два пересекающихся периода могут быть объединены в один более длинный. Определите функцию combine :: Period -> Period -> Period.

Если список Василия: [(10, 20), (40, 50)], а список Татьяны: [(20, 30), (60, 70)], то можно создать общий список периодов, когда один из них занят: [(10, 30), (40, 50), (60, 70)]. Определите функцию combineDiaries :: Diary -> Diary -> Diary, которая это делает.

Составим базу данных из имен людей и их списков дел:

type Person = String

type DB = [(Person, Diary)]

db = [("Василий", [(10, 20), (40, 50)]),

("Татьяна", [(20, 30), (60, 70)])]

Определите функцию meeting :: Int -> Time -> [Person] -> DB -> Period, которая возвращает по списку людей [Person] из базы DB самый ранний свободный период длиной Int минут начиная со времени Time, когда никто из людей не занят. Например: meeting 5 10 ["Василий", "Татьяна"] db == (31, 35).

Напишите программу, которая бы реализовывала описанную выше функциональность, хранила списки дел людей в отдельных файлах в удобочитаемом текстовом виде, позволяла в интерактивном режиме выбрать группу людей и сформировать для них список отрезков их общего свободного времени, а так же назначить совещание.

**Вариант 5. Монеты.**

Создайте программу, которая будет выполнять функции расчета сдачи по заданному номиналу и количеству монет.

Набор имеющихся монет можно представить списком: [1, 1, 5, 5, 5, 10, 10]. Имея их можно заплатить 16 р., но нельзя заплатить 18 р. Представим, что продавец так же имеет монеты. Например: [1, 1, 1, 5]. Тогда 18 р. можно заплатить, отдав 20 р. и получив сдачу 2 р. Функции должны будут определить, может ли покупатель заплатить продавцу заданную сумму со сдачей или без.

Определите функцию subsets :: [a] -> [[a]], которая генерирует все подмножества элементов заданного списка.

Определите функцию amounts :: [Int] -> [Int], которая по набору имеющихся монет может определить все суммы, которые можно заплатить без сдачи.

Определите функцию withChange :: [Int] -> [Int] -> [Int], которая по наборам монет продавца и покупателя может определить все суммы, которые можно заплатить со сдачей. Список сумм не должен содержать отрицательных чисел.

Функция subsets неэффективна, поскольку не принимает во внимание тот факт, что в наборе могут быть монеты одного достоинства. Например: subsets [1, 1] == [[1], [1], [1,1]], где два подмножества относятся к разным монетам одинакового достоинства 1 р. Поскольку нам все равно, какими именно монетами платить, не нужно рассматривать эти подмножества отдельно.

Для улучшения эффективности можно работать с мультимножествами (хранить номинал и количество монет), тогда множество [1, 1, 5, 5, 5, 10, 10] представляется в виде [(1, 2), (5, 3), (10, 2)].

Определите функцию bag :: Eq a => [a] -> [(a, Int)], которая преобразует список в мультимножество.

Определите функцию set :: Eq a => [(a, Int)] -> [a], которая преобразует мультимножество в список.

Определите функцию subBags :: [(a, Int)] -> [[(a, Int)]], которая генерирует все подмультимножества элементов заданного мультимножества. Например:subBags [(1, 2)] == [[], [(1, 1)], [(1, 2)]]

Переопределите функции amounts и withChange для работы с мультимножествами.

Предусмотрите возможность хранения монет продавца и покупателя в файлах, а так же обеспечьте работу с программой в интерактивном режиме.

**Вариант 6. Крестики-нолики.**

Игра «крестики-нолики» на доске 3 х 3. Пронумеруем клетки следующим образом:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Тогда позиция во время игры может быть представлена как data Pos = Pos [Integer] [Integer], где первый список содержит позиции текущего игрока, а второй — позиции противника. Например, если сейчас играет «крестик», тогда позиция Pos [1,5] [2] может быть представлена:

|  |  |  |
| --- | --- | --- |
| x | o |  |
|  | x |  |
|  |  |  |

Определите функцию validPos :: Pos -> Bool, которая проверяет необходимые условия правильности позиции.

Определите функцию vacant :: Pos -> [Integer], которая возвращает пустые клетки для данной позиции. Например: vacant (Pos [1,5] [2]) == [3,4,6,7,8,9].

Определите функцию play :: Integer -> Pos -> Pos, которая возвращает позицию, где текущий игрок ставит свой знак в заданную клетку. Например: play 1 (Pos [5] [2]) == Pos [1,5] [2].

Определите функцию won :: Pos -> Bool, которая определяет, выиграл ли текущий игрок. Например: won (Pos [1,5] [2,3]) == False; won (Pos [1,4,5,7] [2,3,9]) == True.

Сопоставим каждой позиции счет по следующему принципу (считаем, что сейчас — ход текущего игрока):

* Счет равен 1 если текущий игрок может точно победить независимо от хода противника.
* Счет равен -1 если противник может точно победить независимо от хода текущего игрока.
* Счет равен 0 если ни один игрок не может победить.

Счет может быть подсчитан рекурсивно:

* Счет вычисляется сразу же, если кто-нибудь из игроков уже выиграл, или больше нет свободных клеток.
* Иначе рассмотрите все возможные ходы текущего игрока и оцените результирующую позицию *с точки зрения противника*. Текущий игрок должен выбрать ход, ведущий к наихудшей позиции для противника. Если счет противника -1, то счет текущей позиции +1 и наоборот.

Определите функцию score :: Pos -> Integer, определяющую счет текущей позиции. Например: score (Pos [1,5] [2,3]) == 1.

Определите функцию bestMove :: Pos -> Integer, Возвращающую клетку в которую нужно сделать лучший ход. Например: bestMove (Pos [1,5] [2,3]) == 4.

Напишите программу, которая бы позволяла бы пользователю выбирать свой знак, а так же в интерактивном режиме играть с человеком, отображая изменение состояния игрового поля.

Примечание. Представленный здесь минимаксный алгоритм является достаточно медленным, и для реальных игр используется оптимизированный вариант данного алгоритма.

**Вариант 7. Архиватор.**

Для сжатия данных используем кодирование Хаффмена, алгоритм сжатия данных, который используется, в том числе и как часть алгоритма сжатия изображений JPEG. Кодирование Хаффмена основано на анализе входного потока и присваивании каждому символу кода (последовательности бит) так, чтобы наиболее частые символы имели короткие коды, а те, что встречаются реже — более длинные. В выходном потоке каждый символ заменяется своим кодом, что обычно приводит к сокращению объема потока. Например, если необходимо сжать слово «вата», коды можно назначить следующим образом:

|  |  |
| --- | --- |
| а   в   т | 0   10  11 |

Сжатый текст будет следующим:100110, т.е. 6 бит вместо 32 при обычном хранении. (На практике необходимо так же хранить таблицу кодов, поэтому кодирование Хаффмена более эффективно на больших объемах данных).

Коды находятся построением дерева, в котором располагаются все символы текста:

0/\1

/ \

а 0/\1

/ \

в т

Код каждого символа вычисляется как путь от корня дерева до этого символа. Все переходы влево маркируются 0, а вправо — 1. более частые символы располагаются выше в дереве и получают более короткие коды.

Для построения дерева сначала построим тривиальное дерево (содержащее только один символ) для каждого символа входного потока, присвоив ему вес, равный количеству вхождений данного символа во входной поток. Например, строим

* дерево, содержащее только «в» с весом 1,
* дерево, содержащее только «а» с весом 2,
* дерево, содержащее только «т» с весом 1.

Затем соединяем два дерева с *минимальными* весами в одно, присваивая ему суммарный вес исходных деревьев. Например, объединяя деревья «в» и «т» получаем дерево с весом 2:

0/\1

/ \

в т

Данный шаг повторяется до тех пор, пока не останется одно дерево с весом, равным общему количеству символов в файле. Коды Хаффмена можно считать из полученного дерева.

Представим такие деревья с помощью типа data Huffman = Leaf Char | Branch Huffman Huffman. Дерево из нашего примера будет иметь вид: Branch (Leaf 'а') (Branch (Leaf 'в') (Leaf 'т')). Следует отметить, что 0 и 1 хранить необязательно, поскольку они жестко привязаны к левым и правым ветвям.

Представим таблицы кодов с помощью типа type CodeTable = [(Char, String)], например: exampleCodeTable = [('а', "0"), ('в', "10"), ('т', "11")].

Определите функцию encode :: CodeTable -> String -> String, которая кодирует текст, заменяя символы их кодами. Например: encode exampleCodeTable "вата" == "100110".

Определите функцию extractCodes :: Huffman -> CodeTable, которая преобразует дерево Хаффмена в соответствующую таблицу кодов.

Определите функцию buildTree :: [(Int, Huffman)] -> Huffman, которая по списку частичных деревьев получает полное дерево, объединяя по два дерева с наименьшими весами.

Определите функцию huffmanCode :: String -> CodeTable, которая строит таблицу кодов по данной строке.

Определите функцию compress, которая сжимает строку по методу Хаффмэна. На выходе функции должна быть закодированная битовая последовательность и дерево или таблица кодов.

Определите функцию expand, которая по корректной закодированной битовой последовательности и дереву или таблице кодов разжимает строку по методу Хаффмэна.

Напишите программу, которая бы сжимала и распаковывала произвольный файл. Имена входного и выходного файла и режим работы должны считываться из командной строки. Выходной текстовый файл должен содержать таблицу или дерево Хаффмена в компактном виде, а так же выходную битовую последовательность (в текстовом виде).

**Вариант 8. Быки и коровы.**

В данной работе необходимо разработать программу для игры «Быки и коровы». Правила игры: один игрок выбирает четырехзначный код, каждая цифра должна быть от 1 до 6 (например, [5, 2, 2, 1]). Другой игрок должен получить код с помощью нескольких догадок такого же вида (например, [2, 2, 3, 2]). Первый игрок сообщает количество быков и коров для каждой догадки. Бык — это правильная цифра на правильном месте. Корова — это правильная цифра на неправильном месте. Одна и та же цифра может быть посчитана только один раз — как бык или как корова. Зная эту информацию второй игрок пытается получить код за минимальное число попыток.

Определите функцию bulls :: [Int] -> [Int] -> Int, которая по секретному коду и догадке возвращает количество быков:

bulls [1, 2, 3, 4] [1, 1, 2, 2] == 1

bulls [1, 2, 3, 4] [2, 1, 2, 1] == 0

bulls [1, 1, 2, 2] [1, 2, 1, 2] == 2

Определите функцию cows :: [Int] -> [Int] -> Int, которая по секретному коду и догадке возвращает количество коров:

cows [4, 3, 2, 1] [1, 1, 1, 1] == 0

cows [4, 3, 2, 1] [1, 2, 2, 2] == 1

cows [1, 2, 2, 1] [1, 2, 1, 2] == 2

Возможно будет удобно определить вспомогательную функцию, которая вычисляет количество правильных цифр вообще (т.е. быков и коров вместе), а затем вычесть из этого числа количество быков. Помните, что каждая цифра в коде учитывается только один раз.

Определите game :: IO (), которая играет в качестве второго игрока против человека, который загадывает код. Напишите интерактивную программу, которая выдает догадки, считывает счет, который вводится человеком и сообщает код, когда он становится ясным. Пример диалога, если человек загадал [4, 3, 2, 1] (то, что вводил человек — *курсивом*):

Main>*game*

Предположение: [1, 1, 1, 1]. Каков счет? *(1, 0)*

Предположение: [1, 2, 2, 2]. Каков счет? *(1, 1)*

Предположение: [3, 1, 2, 3]. Каков счет? *(1, 2)*

Предположение: [3, 2, 1, 4]. Каков счет? *(0, 4)*

Предположение: [4, 1, 3, 2]. Каков счет? *(1, 3)*

Секретный код: [4, 3, 2, 1].

Для алгоритма можно использовать следующую простую стратегию: создайте список всех возможных кодов, предложите первый из них, после получения счета удалите из списка все варианты, которые не соответствуют счету. Продолжайте итерации до тех пор, пока в списке не останется один код, который можно вывести в качестве угаданного.

**Вариант 9. Раскраска карты.**

Напишите программу, которая выбирает цвета для раскраски стран на карте таким образом, чтобы ни у каких соседних стран цвета не совпадали.

Известно, что для любой плоской карты можно найти такую раскраску с использованием не более четырех цветов.

Представим страны и цвета с помощью строк и определим:

type Color = String

type Country = String

colors = ["красный", "желтый", "синий", "зеленый"]

countries = ["Россия", "Украина", "Белоруссия", "Польша", "Германия", "Франция", "Aвстрия", "Монголия", "Китай"]

Определите функцию neighbors :: Country -> [Country], которая для данной страны возвращает список стран, с которыми она имеет общие границы. Функция должна использовать структуру для хранения взаимного расположения стран (например в виде графа). В этой функции кодируется карта, а остальные функции должны работать с картой только лишь посредством функции neighbors и спискаcountries. Остальной код должен работать даже если эти определения изменятся для представления другой карты.

Представим раскраску стран в виде списка type Coloring = [(Country, Color)]. Например: [("Украина", "синий"), ("Польша", "зеленый")].

Определите функцию safeColor :: Coloring -> Country -> Color -> Bool, которая проверяет, можно ли при имеющейся раскраске окрасить заданную страну в заданный цвет так, чтобы у соседних стран были разные цвета. Например: safeColor [("Украина", "синий"), ("Польша", "зеленый")] "Белоруссия" "красный" == True.

Определите функцию color :: [Country] -> [Coloring], которая возвращает список всех возможных правильных раскрасок стран из данного списка.

Напишите программу, реализующую данную функциональность, предусмотрев ввод исходных данных о цветах, странах и карте из текстовых файлов и вывод возможных раскрасок отдельными строками в текстовый файл.

**Вариант 10. Обратная польская запись.**

Напишите программу, которая будет обрабатывать выражения представленные в обратной польской записи. При этом операции записываются после операндов, например (1 + 2) \* 3 записывается в виде 1 2 + 3 \*. преимуществом данной записи является отсутствие скобок, поскольку порядок выполнения операций задан в самой записи: 1 + (2 \* 3) == 1 2 3 \* +

Ограничимся выражениями, содержащими целые числа и операции +, -, \*, / (т.е. div). Они могут быть представлены в виде типа: data Expr = Num Int | Add Expr Expr | Sub Expr Expr | Mul Expr Expr | Div Expr Expr.

Определите функцию rp :: Expr -> [String], которая переводит выражение в обратную польскую запись. Например: rp (Add (Mul (Num 2) (Num 10)) (Num 3)) == ["2", "10", "\*", "3", "+"].

Для вычисления выражения в обратной польской записи можно использовать стек:

* Каждое число добавляется в вершину стека.
* Операция извлекает два верхних элемента стека, производится над ними и помещает результат в вершину стека.

Определите функцию evaluate :: [String] -> [Int] -> [Int], которая по выражению, представленному в обратной польской записи и начальному содержимому стека возвращает состояние стека после вычисления выражения:

Main> evaluate ["1", "2", "3", "\*", "+"] []

[7]

Main> evaluate ["3", "\*"] [2, 1]

[6, 1]

Напишите программу, реализующую указанную функциональность. Обеспечьте считывание выражений в обратной польской записи из текстового файла и вывод результатов в текстовый файл.

**Вариант 11. Поиск файлов.**

Операционная система рассматривает имена файлов, содержащие \* или ? как шаблоны, где ?представляет один любой символ, \* — любую последовательность символов (включая и пустую), а любой другой символ представляет сам себя. Например: \*.hs соответствует Main.hs, A.hs.hs, а шаблон \*a?c\* соответствует abcarc (двумя способами), но не соответствует back, car, abbc.

Определите функцию matches :: String -> String -> Bool, которая определяет, соответствует ли строка шаблону.

Определите функцию findMatches :: String -> [String] -> [String], которая возвращает те строки из списка, которые соответствуют шаблону.

Определите findFiles :: String -> String -> IO [String], которая возвращает имена всех файлов и каталога, которые соответствуют шаблону. Для получения содержимого каталога можно использовать getDirectoryContents :: String -> IO [String] из модуля Directory, которая возвращает список имен файлов из указанного каталога.

Реализуйте вариант этой функции, который возвращает соответствующие шаблону имена всех файлов из всех подкаталогов данного каталога. Имя файла должно предваряться полным или относительным путем.

**Вариант 12. Система хранения версий.**

Разработайте программу для компактного хранения нескольких версий одного и того же файла. Идея заключается в том, чтобы хранить различия между версиями, которые предположительно будут небольшими.

Представим различия между двумя строками как список команд простейшего редактора:

* *вставить* данный символ в текущую позицию,
* *удалить* символ в текущей позиции,
* *копировать* символ из текущей позиции в выходной поток и передвинуть позицию на один символ вправо.

Редактирование начинается с начала строки и заканчивается, когда позиция редактирования передвинется до конца строки (т.е. обработана вся входная строка, скопирована или удалена).

Например, чтобы получить hello world из helo wosld используются команды:

|  |  |  |
| --- | --- | --- |
| **На входе** | **Команда** | **На выходе** |
| "helo wosld" "elo wosld" "lo wosld" "o wosld" "o wosld" " wosld" "wosld" "osld" "sld" "sld" "ld" "d" "" | копировать копировать копировать вставить l копировать копировать копировать копировать вставить r удалить копировать копировать | "" "h" "he" "hel" "hell" "hello" "hello " "hello w" "hello wo" "hello wor" "hello wor" "hello worl" "hello world" |

Определите тип данных для представления команд редактирования: data Command = ...

Определите функцию edit :: [Command] -> String -> String, которая возвращает строку, полученную обработкой данной строки заданным набором команд.

Список команд не является экономичным способом хранения новых версий, поскольку на каждый входной символ должна быть как минимум одна команда: копирования или удаления. Но когда входной и выходной потоки похожи, то поток команд содержит последовательности повторяющихся команд копирования или удаления. Сожмем последовательность команд по алгоритму RLE, заменив последовательность повторяющихся команд на саму команду и количество ее повторений.

Определите функцию, которая таким образом сжимает и разжимает список: compress :: Eq a => [a] -> [(Int, a)]. uncompress :: [(Int, a)] -> [a].

Определите функцию shorter :: [a] -> [a] -> [a], которая самый короткий из двух переданных ей списков.

Определите функцию shortestEdit :: String -> String -> [Command], которая возвращает кратчайший список команд редактирования, преобразующих первую строку ко второй.

Если одна из строк-аргументов пустая, то существует только один возможный список команд. Если обе строки начинаются с одинакового символа, нужно скопировать его и так же обработать оставшиеся части строк. Если строки начинаются с разных символов, то можно или вставить символ в первую строку, или удалить из второй. Какой способ выбрать? Тот, который в итоге приводит к списку команд наименьшей длины.

Напишите программу, которая позволяла бы получать последовательности команд на основе двух файлов и хранить их в виде текстовых файлов, а также генерировать файлы версий на основе некоторого эталонного текстового файла и файла с последовательностью команд. Режим работы и имена входных файлов должны вводиться из командной строки.