Практическое занятие «Хаскелл–5. Кортежи, алгебраические типы данных»

19 ноября 2019 года

1. Напишите функцию `myZip :: ([a],[b]) -> [(a,b)]` (без использования стандартной функции `zip`), принимающую на вход пару списков и выдающей список из пар, объединяющих соответствующие элементы списков. Длина результат должна быть равна минимуму длин исходных списков. Напишите два варианта этой функции: с использованием рекурсивных вызовов и с использованием функций высших порядков (но не `zip`!). Рассмотрите возможность использования функции `uncurry`.

2. Пусть задан текст в виде списка строк. Напишите функцию
   `freqs :: [String] -> ([Char,Int])`,
   которая формирует частотную гистограмму букв латиницы в этом тексте, то есть список пар (*символ*,*частота*), упорядоченный по убыванию частот. Под *частотой* символа подразумевается количество его вхождений в слова текста. Следует учитывать, что в исходном тексте могут быть как заглавные буквы латинского алфавита, так и строчные, а кроме того, могут быть и другие символы (цифры, знаки пунктуации, пробелы и др.). Потому следует обратить внимание на функции смены регистра символа из библиотеки `Data.Char`. Для фильтрации символов латиницы следует иметь в виду, что они составляют два непрерывных возрастающих интервала в машинном алфавите: от `'A'` до `'Z'` и от `'a'` до `'z'`.

3. Определите тип данных, представляющий информацию о карте в карточной игре. Каждая карта характеризуется одной из четырёх мастей — пики (spades), трефы (clubs), бубны (diamonds), червы (hearts). Карта может быть либо младшей (от двойки до десятки), либо онёром (то есть картинкой: валет — jack, дама — queen, король — king, туз — ace). Определите типы `Suit` для масти, `Value` для достоинства карты, `Card` для описания карты. Затем определите:

   а) функцию `isMinor`, проверяющая, что её аргумент является младшей картой.

   б) функцию `cardsMinMax :: [Card] -> (Card,Card)`, возвращающую минимальную и максимальную среди карт списка; порядки мастей и карт в масти считаем те, которые описаны в начале задачи.

   в) функцию `sameSuit`, проверяющая, что две переданные в неё карты одной масти.

   г) функцию `beats :: Card -> Card -> Bool`, проверяющая, что карта, переданная ей в качестве первого аргумента, бьёт карту, являющуюся вторым аргументом. Карта может быть побита только картой той же масти, имеющей большее старшинство.

   д) функцию `beatsTrump :: Card -> Card -> Suit -> Bool`, аналогичная `beats`, но принимающая в качестве дополнительного аргумента козырную масть. Соответственно, в дополнение к правилу из предыдущей задачи следует учесть, что любая козырная карта бьёт любую некозырную.

   е) функцию `beatsList`, принимающая в качестве аргументов список карт, карту и козырную масть и возвращающая список тех карт из первого аргумента, которые бьют указанную карту с учётом козырной масти.

   ж) функцию `totalValue :: [Card] -> [Int]`, по заданному списку карт возвращающая список чисел, каждое из которых является возможной суммой очков указанных карт: младшие карты считаются по номиналу; валет, дама и король могут считаться за 10 очков, а могут за 2, 3 и 4 очка соответственно; стоимость туза может рассматриваться и как 1, и как 11 очков. Функция должна вернуть список всех возможных вариантов суммы без повторений.