Лабораторная работа № 3.

1. Реализовать консольный интерфейс для третьей задачи из 2-ой лабораторной работы, используя монаду IO и do нотацию. Интерфейс должен давать возможность выполнить все реализованные функции. Программа должна завершаться только после того как пользователь выберет ответствующую опцию, то есть должна быть возможность выполнить несколько функций до завершения программы.
2. Используя монаду список и do нотацию, реализовать функцию, которая будет возвращать все возможные комбинации результатов подбрасывания монеты после n бросков, в которых орел выпадал чаще чем решка. Подсказка: будет удобнее, если реализовать следующие функции.
   2.1. nextCoinToss :: [Coin] -> [[Coin]] - функция принимает в качестве параметра список результатов предыдущих бросков.
   2.2. nextCoinTossN :: [Coin] -> Int -> [[Coin]] - функция принимает в качестве параметра список результатов предыдущих бросков, и число сколько еще нужно сделать бросков.
3. С помощью монады Writer и do нотации написать функцию, которая будет имитировать сдачу сессии студентом, то есть монада Writer должна уметь сохранять в себе список предметов в порядке их сдаче и сумму всех баллов за экзамены. Реализовать функции, которые будут из монады Writer доставать список сданных предметов в порядке их сдачи, и средний балл за все экзамены. Подсказка: в монаде Writer в качестве типа данных для логирования можно использовать кортеж, будет удобнее, если реализовать следующие функции.
   3.1. session :: Writer ([String], ?) () - эта функция будет имитировать сдачу сессии.
   3.2. passExam :: String → Int → Writer ([String], ?) () - эта функция принимает в качестве параметров название предмета и оценку за него.

3.3. getObejects :: Writer ([String], ?) () → [String] – функция возвращает список сданных предметов.

3.4. getAverageScore :: Writer ([String], ?) () → Int - функция возвращает средний балл за сессию.

На месте ? нужно использовать какой-то тип данных, подумать какой. Балльная система может быть от 2 до 5, или например от 0 до 100.

4. Есть некая система в которой зарегистрировались N преподавателей и M студентов по очереди (N и M известны и фиксированы). Есть две функции getStudent, getTeacher, которые могут по номеру k вернуть фамилию студента или преподавателя, k принимает значения от 0 до M или N соответственно. При регистрации пользователь указывал номер телефона, и есть функция, которая по номеру i, где i от 0 до N+M, может вернуть номер телефона пользователя. Используя монаду Reader и do нотацию, реализовать функцию которая по номеру i вернет информацию от том, что пользователь является студентом или преподавателем, и вернет его фамилию и номер телефона. Подсказка: нужно использовать функцию ask, чтобы получить информацию о контексте монады Reader (об аргументе), реализовать следующие функции.
   4.1. getStudent :: Int → String – параметр принимает значение от 0 до M.
   4.2. getTeacher :: Int → String – параметр принимает значение от 0 до N.
   4.3. getInfo :: Int → String – параметр принимает значение от 0 до N + M, и возвращается строка вида: «4. Teacher: Petrov, phone: 89001110022»
5. Используя монаду State и do нотацию, реализовать генератор пароля из 6 символов, используя последовательность псевдослучайных символов из заданного алфавита. Подсказка: реализовать следующие функции.
   5.1. alphabet :: String — список допустимых символов.

5.2. myRandom :: Integer -> (Char, Integer) — функция, которая на основе состояния вычисляет новое состояние и допустимый символ и алфавита alphabet.

5.3. myRandomState :: State Integer Char — функция, которая оборачивает генератор в монаду State.

5.4. genPassord :: State Integer String – функция, которая генерирует пароль из 6 символов, на основе начального состояния.

Требования: Для всех реализованных функций должны быть примеры, демонстрирующие работу.

Ограничения: Во всех задачах необходимо использовать либо do нотацию, либо монадические функции.

Варианты

1) 1, 4
2) 1, 3
3) 1, 3
4) 1, 4
5) 1, 4
6) 1, 2
7) 1, 2
8) 1, 4
9) 1, 5
10) 1, 3
11) 1, 4
12) 1, 3
13) 1, 3
14) 1, 2
15) 1, 5
16) 1, 2
17) 1, 3
18) 1, 2
19) 1, 5

20) 1, 4
21) 1, 5
22) 1, 5
23) 1, 5
24) 1, 5
25) 1, 5
26) 1, 2
27) 1, 3
28) 1, 5
29) 1, 5
30) 1, 2
31) 1, 2
32) 1, 2
33) 1, 4
34) 1, 4
35) 1, 3
36) 1, 4
37) 1, 4
38) 1, 4
39) 1, 4
40) 1, 3
41) 1, 5
42) 1, 3
43) 1, 4
44) 1, 5
45) 1, 2
46) 1, 3
47) 1, 5
48) 1, 5
49) 1, 2
50) 1, 3
51) 1, 4
52) 1, 3
53) 1, 3

54) 1, 2
55) 1, 2
56) 1, 2
57) 1, 5
58) 1, 2
59) 1, 4
60) 1, 3