Контролна работа № 1 по Функционално програмиране
Специалност „Информационни системи“, I курс, 10.04.2021 г.

Задача 1

Едно естествено число се нарича „четновато“ (evenly), ако броят на делителите му (вкл. 1 и самото число) е четно число. В частност, първите три „четновати“ числа са 2 (делителите са 1 и 2 или общо 2 на брой), 3 (делителите са 1 и 3 или общо 2 на брой) и 5 (делителите са 1 и 5 или общо 2 на брой).
Да се дефинира функция `sumOfEvenly :: Int -> Int -> Int`, която да намира сумата от всички „четновати“ числа в интервала [`a`,`b`], където `a` и `b` са цели числа и 1 <= `a` <= `b` <= 1000.

Пример:

```
sumOfEvenly 1 10 ⟶ 41     -- 2+3+5+6+7+8+10
sumOfEvenly 5 20 ⟶ 175    -- 5+6+7+8+10+11+12+13+14+15+17+18+19+20
```

Задача 2

Да се дефинира функция `kthMaxMin :: [Int] -> (Int -> Int)`, която приема списък от цели числа и връща функция с параметър естествено число `k` – такава, че оценката на израза `((kthMaxMin xs) k)` е `k`-тото по абсолютна стойност отрицателно число в `xs`. Ако такова число не съществува, да се връща грешката “No such number”.

Примери:

```
(kthMaxMin [-1]) 1                      ⟶ -1
(kthMaxMin [-1,0,-1,0,-2,3,1,-1]) 3     ⟶ error “No such number”
(kthMaxMin [-1,-5,-6,-6,-6,-6]) 2       ⟶ -5
(kthMaxMin [1,2,3,4,-5,6,7,-2,-1,0]) 2 ⟶ -2
```

Задача 3

Да се дефинира функция `persistence :: Int -> (Int, [Int])`, която приема естествено число `n` и връща наредена двойка от вида `(x,ys)`. Вторият елемент на резултата трябва да е списък `ys`, чийто първи елемент е равен на произведението на цифрите на `n`, а всеки следващ елемент на `ys` е равен на произведението на цифрите на предходния до получаването на едноцифрено произведение, на което е равен последният елемент на `ys`. Първият елемент на резултата е равен на дължината `x` на списъка `ys`. Намирането на произведението на цифрите на дадено число да се реализира с **линейно итеративен процес.**

Примери:

```
persistence 39  ⟶ (3,[27,14,4])       -- 3*9=27, 2*7=14, 1*4=4
persistence 999 ⟶ (4,[729,126,12,2]) -- 9*9*9=729, 7*2*9=126,
persistence 126 ⟶ (2,[12,2])          -- 1*2*6=12, 1*2=2
persistence 4   ⟶ (1,[4])
```

Задача 4

Да се дефинира функция `scoreHand :: [String] -> Int`, която оценява ръка от играта Блекджек (или 21). Функцията приема списък от символни низове, съответни на отделните карти от ръката `("2", "3", ... , "10", "J", "Q", "K", "A")`, и намира общия брой точки (оценката) на ръката.

Правилата на точкуване са следните:

- номерираните карти се оценяват с тяхната стойност (т.е. с число от 2 до 10);
- вале, дама и поп `("J", "Q", "K")` се оценяват с 10;
- асото може да се оцени с 1 или 11.

Функцията трябва да връща оценката на дадена ръка, която е равна на възможно най-голямата сума k от точките на ръката, която е по-малка или равна на 21. Ако такова k не съществува, да се върне възможно най-малката сума от точките на ръката, която е по-голяма от 21.

Примери:

```
scoreHand ["A"]                          → 11
scoreHand ["A","J"]                      → 21
scoreHand ["5","3","7"]                  → 15
scoreHand ["5","4","3","2","A","K"]      → 25
scoreHand ["2","3"]                      → 5
scoreHand ["4","5","6"]                  → 15
scoreHand ["7","7","8"]                  → 22
scoreHand ["K","J","Q"]                  → 30
scoreHand ["A","3"]                      → 14
scoreHand ["A","A"]                      → 12
scoreHand ["A", "10", "A"]               → 12
scoreHand ["A","2","A","9","9"]          → 22
```