Zadanie 4.1

Utwórz moduł **Lists** i zdefiniuj w nim funkcję **sumOfSquares**, która jako argument przyjmuje listę liczb i zwraca sumę ich kwadratów. Funkcja zwraca wartość 0 jeśli lista jest pusta. Zdefiniuj funkcję na dwa sposoby:

1. korzystając z funkcji standardowych (**sum** i **map**);
2. rekurencyjnie.

Zadanie 4.2

W module **Lists** zdefiniuj:

1. funkcję **sum1**, która jako argument przyjmuje listę liczb i zwraca sumę elementów o indeksach nieparzystych;
2. (*) funkcję **sum2**, która jako argument przyjmuje listę liczb i zwraca sumę elementów o indeksach parzystych;
3. (*) funkcję **sum3**, która jako argument przyjmuje listę liczb i zwraca sumę elementów o indeksach 3, 6, 9, …;

Wszystkie funkcje zdefiniuj jako polimorficzne.

Zadanie 4.3

1. Zdefiniuj funkcję **countLower** wyznaczającą liczbę małych liter w przekazanym jej tekście wejściowym.
2. (*) Zdefiniuj funkcję **countLowerUpper**, wyznaczającą liczbę małych i wielkich liter w przekazanym jej tekście wejściowym. Funkcja powinna być następującego typu `countLowerUpper :: String → (Int, Int)`.

Funkcje umieść w module **Lists**.

Zadanie 4.4

(*) Zdefiniuj funkcję **string2bools**, która listę znaków zastępuje listą wartości logicznych określających, czy oryginalny element był małą literą, czy też nie. Funkcję umieść w module **Lists**.

Wskazówka: Do modułu **Lists** poniżej słowa **where** dodaj poniższą linię:

```
import Data.Char
```

Zadanie 4.5

W module **lists** zdefiniuj funkcje, które jako argumenty przyjmują liczbę x i listę liczb tego samego typu i które jako wynik zwracają:

1. funkcja **cgtx** liczbę elementów listy większych od x;
2. (*) funkcja **cltx** liczbę elementów listy mniejszych od x;
3. funkcja **gtx** listę elementów z listy wejściowych większych od x;
4. (*) funckja **ltx** listę elementów z listy wejściowych mniejszych lub równych x.

Zadanie 4.6

Zdefiniuj funkcję **string2int** przekształcającą tekst złożony z cyfr na odpowiednią wartość liczbową. Funkcję umieść w module **Lists**.

Zadanie 4.7

Stosując funkcje standardowe operujące na listach oblicz:

1. suma od 1 do 100 liczb postaci 1/n;
2. iloczyn od 1 do 50 liczb postaci (1+n)/(2+n);
3. (*) suma od 1 do 1000 liczb postaci 1/(i^2);
4. (*) suma od 1 do 1000 liczb postaci (sqrt i)-1/i;
5. (*) iloczyn od 1 do 1000 liczb postaci (i+1)/(i^3).

Zadanie 4.8

W module **Lists** umieść definicje funkcji **factors** i **prime** ze slajdu 45. Przetestuj ich działanie. Przeanalizuj działanie poniższej funkcji:

```
primes :: [Integer]

primes = filter prime [2 ..]
```

Funkcję umieść w module **Lists**. Aby zatrzymać obliczenia użyj kombinacji **Ctrl + C**.

Do modułu dodaj funkcję **pairs** i przeanalizuj jej działanie na przykładowych listach z liczbami całkowitymi:

```
pairs :: [Integer] -> [(Integer, Integer)]

pairs (x:y:[])   | x + 2 == y = [(x,y)]

                 | otherwise  = []

pairs (x:y:xys) | x + 2 == y = (x,y) : pairs (y:xys)

                 | otherwise  = pairs (y:xys)
```

1. (*) Zdefiniuj funkcję **primePairs** zwracającą listę par liczb pierwszych takich, że drugi element pary jest większy o 2 od pierwszego.
2. (*) Zdefiniuj funkcję **primeTriples** zwracającą listę trójek liczb pierwszych takich, że drugi element trójki jest większy o 2 od pierwszego, a trzeci jest większy o dwa od drugiego.