Ćwiczenie 4
Haskell - operacje na listach

1 Wprowadzenie

1.1 Listy

Listy są ciągami wartości określonego typu. Zapis listy:

```
[element_0, element_1, ...]
```

1.2 Notacja przedziałowa

Lista z elementami od a do b z krokiem 1:

```
[a ..  b]
```

Przykład:

```
[1 ..  200]
```

1.3 Wzorce dla list

Przykłady:

- [] - lista pusta,
- (glowa : ogon) - lista co najmniej jednoelementowa z głową i ogonem,
- [_] - lista jednoelementowa z dowolnym elementem,
- [x, y] - lista dwuelementowa,
- [x, _, y] - lista trójelementowa z dowolnym drugim elementem,
- [_, _, _] - lista trójelementowa z dowolnymi elementami,
- (glowa : srodek : ogon) - lista co najmniej trójelementowa z głową, środkiem i ogonem.

1.4 Użyteczne funkcje

Przykładowe funkcje modułu **Data.Char**:

- **isLower**,
- **isUpper**,
- **isDigit**.

Przykładowe funkcje operujące na listach:

- **sum** - funkcja zwracająca sumę elementów listy,
- **length** - funkcja zwracająca liczbę elementów (długość) listy,
- **map** - funkcja stosująca funkcję będącą jej pierwszym argumentem do wszystkich elementów listy będącej jej drugim argumentem,
- **filter** - funkcja usuwająca z listy elementy nie spełniające wskazanego predykatu logicznego.

1.5 Definicja podwyrażeń

Przykład:

```
funkcja1 x = funkcja2 x + x where funkcja2 y = 2*y
```

2 Zadania

2.1

Zdefiniuj moduł z funkcjami:

- *sumaKwadratow*, która jako argument przyjmuje listę liczb i zwraca sumę ich kwadratów. Funkcja zwraca wartość 0 jeśli lista jest pusta.
- *sumaWartosciBezwzglednych*, która jako argument przyjmuje listę liczb i zwraca sumę ich wartości bezwzględnych. Funkcja zwraca wartość 0 jeśli lista jest pusta.

Zdefiniuj powyższe funkcje na dwa sposoby:

- wykorzystując funkcje **sum** i **map**,
- wykorzystując rekurencję.

2.2

Zdefiniuj moduł z funkcjami:

- *sumaParzystych*, która jako argument przyjmuje listę liczb i zwraca sumę elementów tej listy o indeksach parzystych. Funkcja zwraca wartość 0 jeśli lista jest pusta.
- *sumaNieparzystych*, która jako argument przyjmuje listę liczb i zwraca sumę elementów tej listy o indeksach nieparzystych. Funkcja zwraca wartość 0 jeśli lista jest pusta.

2.3

Zdefiniuj moduł z funkcjami:

- *iloscMalych*, która jako argument przyjmuje listę znaków i zwraca liczbę małych liter znajdujących się na liście. Funkcja zwraca wartość 0 jeśli lista jest pusta.
- *iloscDuzych*, która jako argument przyjmuje listę znaków i zwraca liczbę dużych liter znajdujących się na liście. Funkcja zwraca wartość 0 jeśli lista jest pusta.

- *iloscCyfr*, która jako argument przyjmuje listę znaków i zwraca liczbę cyfr znajdujących się na liście. Funkcja zwraca wartość 0 jeśli lista jest pusta.
- *iloscDolarow*, która jako argument przyjmuje listę znaków i zwraca liczbę znaków dolara ($) znajdujących się na liście. Funkcja zwraca wartość 0 jeśli lista jest pusta.
- *iloscLiterOdADoK*, która jako argument przyjmuje listę znaków i zwraca liczbę liter A, B, ..., J, K znajdujących się na liście. Funkcja zwraca wartość 0 jeśli lista jest pusta.

2.4

Zdefiniuj moduł z funkcjami obliczającymi:

- $y = \sum_{i=1}^{100} \frac{i}{i+2}$
- $y = \sum_{j=1}^{50} 2j$
- $y = \sum_{k=1}^{1000} \frac{\sqrt{k}}{2} + 3$

2.5

Zdefiniuj moduł z funkcją zamieniającą listę wartości zero-jedynkowych na listę odpowiadających im wartości true/false.

2.6

Zdefiniuj moduł z funkcją, która jako argumenty przyjmuje listę liczb oraz liczbę x, a zwraca ilość liczb na liście większych od x. Funkcja zwraca wartość 0 jeśli lista jest pusta.

2.7

Zdefiniuj moduł z funkcją, która jako argument przyjmuje listę liczb, a zwraca ilość liczb ujemnych i ilość liczb dodatnich na liście. Funkcja zwraca (0,0) jeśli lista jest pusta.

2.8

Zdefiniuj moduł z funkcją, która jako argument przyjmuje łańcuch znaków, a zwraca ilość małych liter i ilość dużych liter na liście. Funkcja zwraca (0,0) jeśli lista jest pusta.