Ćwiczenie 2
Haskell - definiowanie prostych funkcji

1 Wprowadzenie

1.1 Komentarze

```
-- to jest komentarz jednoliniowy

{- To jest
komentarz wieloliniowy -}
```

1.2 Podstawowe typy danych

- Znakowe: **Char** (pojedynczy znak, np. **'a'**), **String** (łańcuch znaków, np. **"Haskell"**, synonim dla **[Char]**).
- Liczbowe: **Number**, **Int**, **Integer**, **Float**, **Double**.
- Logiczny: **Bool** (wartości logiczne: **True**, **False**).

1.3 Wybrane operatory

- arytmetyczne:

  `+, -, *, /, ^`

- logiczne:

  `&&, ||`

- relacyjne:

  `<, <=, >, >=, ==, /=`

- konkatenacja:

  `++`

Operatory mogą być używane jako funkcje, np.:

```
(*) 2 4
(&&) True False
```

Jako operatorów można używać funkcji dwuargumentowych, np.:

```
4 `mod` 3
7 `div` 2
```

1.4 Definiowanie funkcji

1.4.1 Definiowanie funkcji w trybie interaktywnym

```
let nazwaFunkcji argumenty_wejściowe = wyrażenie
```

Przykład:

```
let dodawanie x y = x+y
```

1.4.2 Definiowanie funkcji w trybie wsadowym

```
nazwaFunkcji argumenty_wejściowe = wyrażenie
```

Przykład:

```
odejmowanie a b = a-b
```

1.4.3 Wywołanie funkcji

```
nazwaFunkcji argumenty
```

Przykłady:

```
dodawanie 2 4
odejmowanie 5 2
```

1.5 Definiowanie modułu

```
module NazwaModułu where
{- definicje funkcji -}
```

1.5.1 Jawne określenie typów dla funkcji

```
nazwaFunkcji :: typy_argumentów_wejściowych -> typ_wartości
nazwaFunkcji argumenty_wejściowe = wyrażenie
```

Przykład:

```
odejmowanie :: Integer -> Integer -> Integer
odejmowanie a b = a-b
```

1.5.2 Sprawdzenie typów dla funkcji

```
:type nazwaFunkcji
```

lub w skrócie

```
:t nazwaFunkcji
```

Przykłady:

```
:type odejmowanie
:t odejmowanie
```

2 Zadania

2.1

W trybie interaktywnym, korzystając z polecenia **let**, zdefiniuj:

- funkcję stałą $f(x) = 5$,
- funkcję liniową $f(x) = 2x + 8$.

Przetestuj działanie funkcji na przykładowych danych.

2.2

Zdefiniuj moduł **Cwiczenie** (zapisz go w pliku **Cwiczenie.hs**). W module umieść:

- funkcję **f1** konkatenującą podany łańcuch sam ze sobą,
- funkcję **f2** zmniejszającą podany argument o 3,
- funkcję **f3** zwracającą resztę z dzielenia podanego argumentu przez 3,
- funkcję **f4** zwracającą wynik porównania podanego argumentu z liczbą 3.
- funkcję **f5** zwracającą część całkowitą z dzielenia podanego argumentu przez 3.

Do każdej funkcji dodaj informację o jej typie oraz komentarz opisujący jej działanie. Przetestuj działanie funkcji na przykładowych danych. Przetestuj działanie polecenia **:t**.

2.3

Zdefiniuj moduł **Matematyka** (zapisz go w pliku **Matematyka.hs**) i umieść w nim definicje wybranych predefiniowanych funkcji matematycznych, ale z polskimi nazwami, np. **pierwiastek**, **logarytm**. Przetestuj działanie funkcji na przykładowych danych.

2.4

Zdefiniuj moduł **Geometria2D** (zapisz go w pliku **Geometria2D.hs**) i umieść w nim definicje funkcji obliczających obwody i pola powierzchni podstawowych figur płaskich (np. prostokąta, trójkąta, koła, równoległoboku). Przetestuj działanie funkcji na przykładowych danych.

2.5

Zdefiniuj moduł **Geometria3D** (zapisz go w pliku **Geometria3D.hs**) i umieść w nim definicje funkcji obliczających pola powierzchni i objętości podstawowych brył (np. prostopadłościanu, ostrosłupa, stożka, kuli). Przetestuj działanie funkcji na przykładowych danych.