Ćwiczenie 6
Haskell - tworzenie plików wykonywalnych, operacje na plikach tekstowych

1 Wprowadzenie

1.1 Kompilacja do pliku wykonywalnego

Kod w Haskell-u może zostać skompilowany do pliku wykonywalnego jeśli nazwą modułu jest **Main** i moduł ten zawiera funkcję **main**.

1.2 Notacja do

Notacja **do** służy do pisania programów w stylu imperatywnym.

1.3 Konwersje

Wybrane funkcje konwersji:

- **show** z klasy **Show** - konwersja wartości odpowiedniego typu na wartość typu **String**.
- **read** z klasy **Read** - konwersja wartości typu **String** na wartość odpowiedniego typu.

1.4 Operacje na listach znaków

Wybrane funkcje operujące na listach znaków:

- **words** - konwersja tekstu (łańcucha znaków) ma listę słów.
- **lines** - konwersja tekstu (łańcucha znaków) ma listę linii.
- **unwords** - konwersja listy słów w tekst (łańcuch znaków) znaków.
- **unlines** - konwersja listy linii w tekst (łańcuch znaków) znaków.

1.5 Operacje wejścia/wyjścia

Wybrane funkcje wykonujące operacje wejścia/wyjścia:

- **print** - wyświetlenie wartości dowolnego typu należącego do klasy **Show** z dodaniem znaku nowej linii.
- **putStr** - wyświetlenie łańcucha znaków.
- **putStrLn** - wyświetlenie łańcucha znaków z dodaniem nowej linii.
- **getLine** - pobranie łańcucha znaków (linii tekstu).

1.6 Operacje na plikach tekstowych

Wybrane funkcje operujące na plikach tekstowych:

- **writeFile**, **appendFile** - zapis do pliku drugiego argument typu **String**.
- **readFile** - odczyt zawartości pliku (zwracana jest wartość typu **String**).

2 Zadania

2.1

Przeanalizuj poniższy kod:

```
module Main where

a :: Int -> Int
a n | n == 0 = 2
 | n > 0 = 2+3*a (n-1)

main = do
  putStrLn "Podaj n: "
  k <- readLn
  print (a k)
```

Co robi ten program? Skompiluj program, a następnie uruchom go w wierszu poleceń.

2.2

Przeanalizuj poniższy kod:

```
module Main where

main = do
 tekst <- readFile "wejscie.txt"
 writeFile "wyjscie.txt"
```

Co robi ten program? Skompiluj program, a następnie uruchom go w wierszu poleceń. Zmodyfikuj kod tak, aby w pliku wejściowym mała litery z pliku wejściowego były zapisane jako duże litery.

2.3

Przeanalizuj poniższy kod:

```
module Main where

import System.Environment

main = do
 [we,wy] <- getArgs
 tekst <- readFile we
 writeFile wy tekst
```

Co robi ten program? Skompiluj program, a następnie uruchom go w wierszu poleceń.

2.4

Plik **dane.txt** ma postać:

```
10 3 4 -9 22 33 0 0 3
```

Przeanalizuj poniższy kod:

```
module Main where

wypisz [] = return ()
wypisz (glowa:ogon) = do putStrLn glowa; wypisz ogon

main = do
 tekst <- readFile "dane.txt"
 let liczby = map read (words tekst)
 wypisz (map show (filter (<=10) liczby))
```

Co robi ten program? Skompiluj program, a następnie uruchom go w wierszu poleceń. Zmodyfikuj kod tak, aby elementy listy wyświetlane były w jednej linii tekstu.