МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №4**

по дисциплине: **«Функциональное программирование»**

на тему: «**Работа в интерпретатора Haskell**»

ВЫПОЛНИЛ студент группы 16 ИТ-3

Яблонский А.С.

ПРОВЕРИЛ преподаватель

Попкова Д.В.

Полоцк, 2018 г.

**Вариант №8**

**Условие варианта:**

Иерархия должностей в некоторой организации образует древовидную структуру. Каждый работник, однозначно характеризующийся уникальным именем, имеет несколько подчиненных. Определите тип данных, представляющий такую иерархию и опишите необходимые функции.

Для реализации была разработана следующая структура данных:

data Stuff =

Stuff {

name :: String,

sub :: [Stuff]

} deriving (Eq, Show)

Для тестирования и отладки пользовалась следующая функция создания иерархии:

hier = Stuff "Ival"[

Stuff "Alexandr" [

Stuff "Don" [

Stuff "Egor" []

]

],

Stuff "Jhon" []

]

**Функция 1**

**getSubordinate()** : возвращающая список подчиненных указанного работника.

**Реализация:**

Для выполнения данного задания была реализована следующая функция:

getSubordinates :: Stuff -> String -> [Stuff]

getSubordinates hs name1 = innerF hs

where innerF x

| name x /= name1 = foldr ((++) . innerF) [] (sub x)

| otherwise = sub x

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Программа отработала, как и ожидалось. Для тестирования функция запускалась со следующими параметрами:

getSubordinates hier "Dons"

**Функция 2**

**getAllSubordinate()** : возвращающая список всех подчиненных данного работника, включая косвенных.

**Реализация:**

Для выполнения данного задания была реализована следующая функция:

getAllSubordinates :: Stuff -> String -> [Stuff]

getAllSubordinates hs = innerF . (getSubordinates hs)

where innerF = concatMap (\x -> x : innerF (sub x))

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Программа отработала, как и ожидалось. Для тестирования функция запускалась со следующими параметрами:

getAllSubordinates hier

**Функция 3**

**getBoss()** : возвращает начальника указанного работника.

**Реализация:**

Для выполнения данного задания была реализована следующая функция:

getBoss :: Stuff -> String -> [Stuff]

getBoss hs name1 = innerF [] hs

where innerF s x

| name x /= name1 = foldr ((++) . innerF [x]) [] (sub x)

| otherwise = s

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Программа отработала, как и ожидалось. Для тестирования функция запускалась со следующими параметрами:

getBoss hier "Dons"

**Функция 4**

**getList()** : возвращающую список пар, первым элементом которых является имя работника, а вторым — количество его подчиненных (включая косвенных).

**Реализация:**

Для выполнения данного задания была реализована следующая функция:

getList :: Stuff -> [(String, Int)]

getList hs = (name hs, length innerF) : innerF

where innerF = concatMap getList (sub hs)

**Тестирование:**

В результате тестирования функции ошибок выявлено не было. Программа отработала, как и ожидалось. Для тестирования функция запускалась со следующими параметрами:

getList hier

**Вывод**

В результате выполнения данной лабораторной работы, приобрел навык работы с методами обработки списком *foldr* и *concatMap.* Также закрепил навыки работы с пользовательскими типами данных и их конструкторами.