**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5

Функциональное и логическое программирование

Выполнил: ст. группы ПВ-21  
Ковалев Павел Александрович

Проверил: Поляков В.К.

**Белгород 2020**

**Вариант 8**

**Задания**

1.Определите следующие функции с использованием функций высшего порядка:

1)Функция вычисления арифметического среднего элементов списка вещественных чисел с использованием функции foldr. Функция должна осуществлять только один проход по списку.

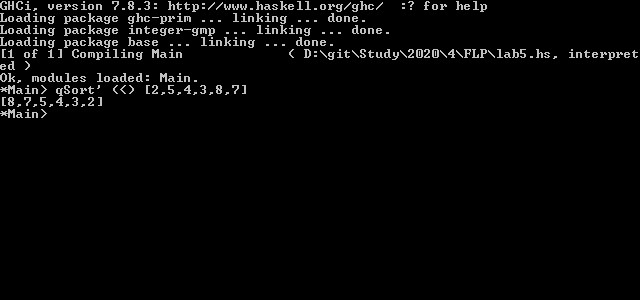
2)Функция, вычисляющая скалярное произведение двух списков (используйте функции foldr и zipWith).

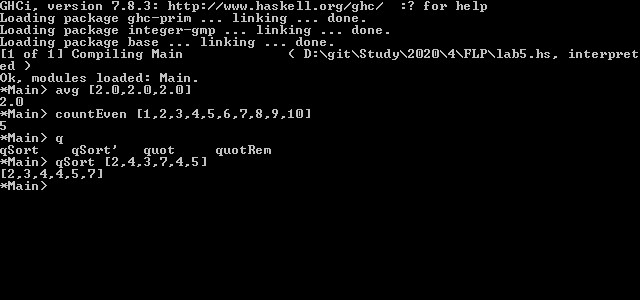
3)Функция countEven, возвращающая количество четных элементов в списке.

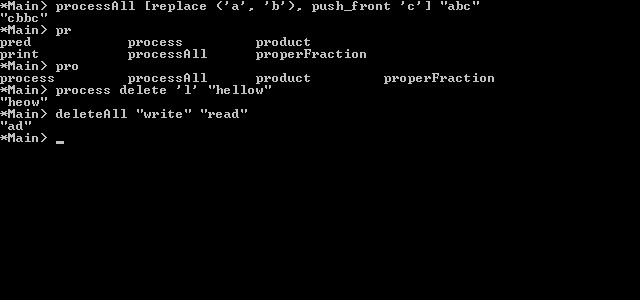
4)Функция quicksort, осуществляющая быструю сортировку списка по следующему рекурсивному алгоритму. Для того чтобы отсортировать список xs, из него выбирается первый элемент (обозначим его x). Остальной список делится на две части: список, состоящий из элементов xs, меньших x и список элементов, больших x. Эти списки сортируются (здесь проявляется рекурсия, поскольку они сортируются этим же алгоритмом), а затем из них составляется результирующий список вида as ++ [x] ++ bs, где as и bs – отсортированные списки меньших и больших элементов соответственно.

5)Определенная в предыдущем пункте функция quicksort сортирует список в порядке возрастания. Обобщите ее: пусть она принимает еще один аргумент – функцию сравнения типа a -> a -> Bool и сортирует список в соответствие с ней.

2.Вернитесь к заданниям из лабораторной работы №3 и реализуйте их с помощью функций высшего порядка. Постарайтесь полностью исключить из определений функций явный проход по списку.





**Задания 2**

**Код программы**

avg :: [Double] -> Double

avg x = (foldr (\ a y -> a+y) 0.0 x) / (fromIntegral $ length x)

scalProd :: [Double] -> [Double] -> Double

scalProd x y = foldr (+) 0.0 (zipWith (\*) x y)

countEven :: [Int] -> Int

countEven = length . filter even

qSort :: Ord a => [a] -> [a]

qSort [] = []

qSort (x:xs) = qSort (filter (< x) xs) ++ [x] ++ qSort (filter (>= x) xs)

qSort' :: (a -> a -> Bool) -> [a] -> [a]

qSort' fc [] = []

qSort' fc (x:xs) = qSort' fc (filter (fc x) xs) ++ [x] ++ (qSort' fc (filter (\ y -> not $ fc x y) xs))

--Rework lab3

--Задание:

--Определим следующий набор операций над строками:

--Очистка: удаление всех символов из строки

--Удаление: удаление всех вхождений указанного символа

--Замена: замена всех вхождений одного символа на другой

--Добавление: добавление в начало строки указанного символа

--Разработайте тип данных, характеризующий операции над строками. Определите следующие функции:

--1)process, получающая в качестве аргумента действие и строку и возвращающая строку, модифицированную в соответствии с указанным действием.

--2)processAll, аналогичная предыдущей, но получающая список действий и выполняющая их по порядку.

--3)deleteAll, принимающая две строки и удаляющая из второй строки все символы первой. При реализации обязательно использовать функцию processAll.

type ProcS = String -> String

clear :: ProcS

clear = const []

delete :: Char -> ProcS

delete = filter . (/=)

replace :: (Char, Char) -> ProcS

replace = map . sub where

sub (c, e) d =

if c == d then e else d

push\_front :: Char -> ProcS

push\_front = (:)

process = ($)

processAll = flip $ foldl $ flip process

deleteAll = processAll . map delete