**კურსის დასახელება:** ფუნქციონალური დაპროგრამება **სტატუსი:** ძირითადი

**ქულების განაწილება:** 1 - **5**ქ., 2 - **5** ქ., 3 - **5** ქ., 4 - **5** ქ., 5 - 1**5** ქ. =34 **ქულათა ჯამი:** **40** ქულა

**გამოცდის ხანგრძლივობა: 2 საათი.**

უი ახლა შევამჩნიე მესამეში /-ის ნაცვლად \*\* ეს უნდა იყოს:

**ძველ: foldr (\*) 1 xs / (1 / fromIntegral length xs**)**)**

**შესწორებული: foldr (\*) 1 xs \*\* (1 / fromIntegral length xs**)**)**

პატივიცემით ირაკლი ნოზაძე

**ბილეთი**

1. სიის კონსტრუქტორის გამოყენებით დაწერეთ mymap სახელწოდების მაღალი რიგის საბიბლიოთეკო ფუნქციის განმარტების თქვენი ვერსია, რომელიც ახორციელებს მითითებულ მოქმედებას (ფუნქციას) სიის ყოველ ელემენტზე . განსაზღვრეთ ფუნქციის ტიპი. მოიყვანეთ გამოძახების მაგალითი.

**map :: (a -> b) -> [a] -> [b] +5**

**mymap changer as = [ changer a | a<-as ]**

**და ადსც შეიძლება:**

**mymap changer as**

**| as==[] = []**

**| otherwise = (changer (head as)) : mymap changer (tail as)**

**mymap (\x -> x+1) [1,2,3] = [2,3,4]**

1. რეკურსიის საშუალებით. დაწერეთ მაღალი რიგის საბიბლიოთეკო myfilter ფუნქცია, რომელიც ირჩევს სიიდან ყოველ ელემენტს, რომელიც პრედიკატს აკმაყოფილებს განსაზღვრეთ ფუნქციის ტიპი. მოიყვანეთ გამოძახების მაგალითი.

**myfilter :: (a -> Bool) -> [a] -> [a] +5**

**myfilter pred [] = []**

**myfilter pred (x:xs)**

**| pred x = x : myfilter xs**

**| otherwise = myfilter xs**

**myfilter (<5) [1,2,3,4,5,6,7,8,9,10]**

**[1,2,3,4,5]**

1. განსაზღვრეთ რეკურსიულად ან სიის კონსტრუქტორის გამოყენებით ფუნქცია, რომელიც შესასვლელზე ღებულობს ნამდვილ რიცხვების სიას და ითვლის ორ მნიშვნელობას: მათ საშუალო არითმეტიკულს და საშუალო გეომეტრიულს. შეეცადეთ, რომ ფუნქციამ მხოლოდ ერთხელ გადახედოს სიას. განსაზღვრეთ ფუნქციის ტიპი. მოიყვანეთ გამოძახების მაგალითი.

**average :: (Num a) => [a] -> a არ მუშაობს +2**

**average xs = sum xs / length xs**

**average [0,1,2,3,4]**

**2**

**geometric :: (Num a) => [a] -> a**

**geometric** **xs = foldr (\*) 1 xs / (1 / fromIntegral length xs**)

**geometric [1,9]**

**3**

**getBoth :: (Num a) => [a] -> (a,b)**

**getBoth xs = (sum xs / length xs, foldr (\*) 1 xs / (1 / fromIntegral length xs**)**)**

1. განსაზღვრეთ maRali rigis *any* da *takeWhile* funqciebi standartul prelude failSi mocemuli maTi aRwerebis gamouyeneblad.

**Any :: (a -> bool) -> [a] -> bool +5**

**Any \_ [] = False**

**Any p xs = or [p x | x <- xs]**

**Any (>5) [0,1,2,3,4]**

**False**

**takeWhile :: (a -> bool) -> [a] -> bool**

**takeWhile f = foldr (\x acc -> if f x then x : acc else []) []**

**takeWhile (<3) [1,2,3,4,5,6]**

**[1,2]**

1. უძრავი ქონების სააგენტოში იყიდება ბინები- Flat, ოთახები- Room და კერძო სახლები-House. ბინა ხასიათდება სართულით, ფართობით და სახლის სართულების რაოდენობით. ოთახი ხასიათდება ამის გარდა კიდევ ფართობით (დამატებით მთელი ბინის ფართობისა). კერძო სახლი ხასიათდება მხოლოდ ფართობით. ანუ განსაზღვრულია მონაცემთა ტიპი, რომელიც წარმოადგენს უძრავი ქონების ობიექტებზე ინფორმაციას. ანუ მოცემული გვაქვს ტიპი:

data NedvObject = Flat Int Int Int | Room Int Int Int Int | House Int deriving (Eq,Show)

-- Flat sarTuli, farTobi, sarTulianoba | Room sarTuli, farTobi, sarTulianob,a oTaxis farTobi | House ფართობი

მონაცემთა ბაზაში ინახება მნიშვნელობების წყვილები, რომელთაგან პირველი წარმოადგენს უძრავ ობიექტს, მეორე–მის ფასს.

განსაზღვრეთ შემდეგი ფუნქციები:

1. GetRoom - მონაცემთა ბაზიდან არჩევს ოთახებს; =17

**GetRoom:: [(NedvObject,Int)] -> [(NedvObject,Int)]**

**GetRoom [] = []**

**GetRoom ((Room x,y):xs) = (Room x,y): GetRoom xs**

**GetRoom (\_:xs) = GetRoom xs**

1. getFlatByPrice - ბაზიდან ირჩევს ბინებს, რომელთა ფასი მოცემულის ტოლია;

**getFlatByPrice:: [(NedvObject,Int)] -> Int -> [(NedvObject,Int)]**

**getFlatByPrice [] \_ = []**

**getFlatByPrice ((Flat a b c,y):xs) price =**

**if y == price then (Flat a b c,y): getFlatByPrice xs price**

**else getFlatByPrice xs price**

1. getExceptBounds, irCevs binebs mocemul sarTulze;

**getByLevel :: [(NedvObject,Int)] -> Int -> [(NedvObject,Int)]**

**getByLevel [] \_ = []**

**getByLevel ((Flat x y z, a) : xs) floor =**

**if x==floor then (Flat x y z, a) : getByLevel xs floor**

**else getByLevel xs floor**

**getByLevel (\_ : xs) floor = getByLevel xs floor**

1. getByType - მეორე არგუმენტად გადაეცემა სტრიქონი, ფასი და ბაზიდან შესაბამის ინფორმაციას იღებს. მაგალითად,

**getByType :: [(NedvObject,Int)] -> String -> [(NedvObject,Int)]**

**getByType [] \_ = []**

**getByType (x:xs) t = case x of**

**(Flat xx y z, a) -> if t=="Flat" then x:getByType xs t**

**else getByType xs t**

**(Room xx y z zz, a) -> if t=="Room" then x:getByType xs t**

**else getByType xs t**

**(House y, a) -> if t=="House" then x : getByType xs t**

**else getByType xs t**

**\*Main>** getByType [(Flat 3 100 10,1000),(Room 4 120 9 20,1000),(House 200,1000),(Flat 1 100 10,900)] "Room" 1001

[(Room 4 120 9 20,1000)]

it :: [(NedvObject, Int)]

it :: [(NedvObject, Int)]