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

**ქულების განაწილება:** 1 - **5**ქ., 2 - **5** ქ., 3 - **5** ქ., 4 - **5** ქ., 5 - 1**5** ქ. =24 **ქულათა ჯამი:** **40** ქულა

**გამოცდის ხანგრძლივობა: 2 საათი.**

**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.

mymap changer as = [ changer a | a<-as ] +5

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 p [] = []

myfilter p (x:xs) | p x = x : filter xs

| otherwise = myfilter xs

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

[4,5,6,7,8,9,10]

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. =0
2. განსაზღვრეთ maRali rigis *all* da *dropWhile* funqciebi standartul prelude failSi mocemuli maTi aRwerebis gamouyeneblad.

// all +5

all p xs = and [ p x | x <- xs]

\*Main> let all p xs = and [ p x | x <- xs]

all :: (t -> Bool) -> [t] -> Bool

\*Main> all even [1,2,3,4]

False

it :: Bool

any p xs = or [ p x | x <- xs]

\*Main> let any p xs = or [ p x | x <- xs]

any :: (t -> Bool) -> [t] -> Bool

\*Main> any even [1,2,3,4]

True

it :: Bool

// dropwhile

სიის კონსტრუქტორით:

Prelude> let dropWhile p xs = [x | x <- xs, not(p x)]

Prelude> dropWhile (<3) [1,2,3,4,5,6,7]

[3,4,5,6,7]

რეკურსიით:

dropWhile p [] = []

dropWhile p (x:xs)

| p x = dropWhile p xs

| otherwise = x:xs

\*Main> dropWhile (<3) [1,2,3,4,5,6,7]

[3,4,5,6,7]

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;
2. getFlatByPrice - ბაზიდან ირჩევს ბინებს, რომელთა ფასი მოცემულის ტოლია;
3. getExceptBounds, irCevs binebs mocemul sarTulze;

getExceptBounds :: [(NedvObject,Int)] -> [(NedvObject,Int)] +4

getExceptBounds [] = []

getExceptBounds ((Flat x y z,a):xs) =

if (x/=z)&&(x/=1) then (Flat x y z,a):getExceptBounds xs

else getExceptBounds xs

getExceptBounds (\_:xs) = getExceptBounds xs

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

**\*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)]

getByType :: [(NedvObject,Int)] -> String -> [(NedvObject,Int)] +5

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