**კურსისდასახელება:**ფუნქციონალური დაპროგრამება **სტატუსი:**ძირითადი

**ქულებისგანაწილება:** 1 -**5**ქ., 2 -**5**ქ., 3 -**5**ქ., 4 -**5**ქ.,5 -1**5**ქ = 22 .**ქულათაჯამი:40**ქულა

**გამოცდის ხანგრძლივობა: 2საათი.**

**ბილეთი**

1. რეკურსიის 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 :: (a -> b) -> [a] -> [b] +5

mymap m [] = []

mymap m (x:xs) = m x : mymap m xs

\*Test> mymap abs[6,20,-36,-5]

[6,20,36,5]

1. სიის კონსტრუქტორის გამოყენებით. 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 \_ [] = []

myfilter f (x:xs)

| f x = x : (myfilter f xs)

| otherwise = myfilter f xs

\*Test> myfilter (>14) [13,14,15,16]

[12,13

1. განსაზღვრეთ რეკურსიულად და სიის კონსტრუქტორის გამოყენებით ფუნქცია, რომელიც შესასვლელზე ღებულობს სიას და აბრუნებს ორ მნიშვნელობას: თავდაპირველი სიიდან ლუწი რიცხვების სიას გაერთიანებულს კენტი რიცხვების სიასთან და დადებითი რიცხვების სიას გაერთიანებულს კენტი რიცხვების სიასთან თავდაპირველ სიაში მათი თანმიმდევრობის შენარჩუნებით. განსაზღვრეთ ფუნქციის ტიპი და მოიყვანეთ გამოძახების მაგალითი. =0
2. gansazRvreT maRali rigis *all*da*dropWhile* funqciebi standartul prelude failSi mocemuli maTi aRwerebis gamouyeneblad.

Fst :: (a ,b) -> a =0

Head :: [a] -> a

Take :: int -> [a] -> [a]

Zip :: [a] -> [b] -> [(a,b)]

Id :: a -> a

1. უძრავიქონებისსააგენტოშიიყიდებაბინები- Flat, ოთახები- Roomდაკერძოსახლები-House. ბინახასიათდებასართულით, ფართობითდასახლისსართულებისრაოდენობით. ოთახიხასიათდებაამისგარდაკიდევფართობით (დამატებითმთელიბინისფართობისა). კერძოსახლიხასიათდებამხოლოდფართობით. ანუ განსაზღვრულია მონაცემთატიპი, რომელიცწარმოადგენსუძრავიქონებისობიექტებზეინფორმაციას. ანუ მოცემული გვაქვს ტიპი:

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

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

მონაცემთაბაზაშიინახებამნიშვნელობებისწყვილები, რომელთაგანპირველიწარმოადგენსუძრავობიექტს, მეორე–მისფასს.

განსაზღვრეთშემდეგიფუნქციები:

1. getFlat monacemTa bazidan irCevs binebs;
2. getRoomByPrice - ბაზიდან ირჩევს ოთახებს, რომელთა ფასი მოცემულზე მეტია;
3. getExceptBounds, ირჩევს მონაცემთა ბაზიდან ბინებს, რომლებიც არ მდებარეობს პირველ და ბოლო სართულებზე.
4. getByType - მეორე არგუმენტად გადაეცემა სტრიქონი 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"

[(Room 4 120 9 20,1000)]

it :: [(NedvObject, Int)]

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

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

data Requirements = Type String | MinArea Int | MaxPrice Int | NeedFloor Int

-- getflat monacemTa bazidan irCevs kerZo saxlebs =12

getflat :: [(NedvObject,Int)] -> [(NedvObject,Int)]

getflat [] = []

getflat ((flat x,y):xs) = (flat x,y): getflat xs

getflat (\_:xs) = getflat xs

-- getRoomByPrice - ბაზიდან ირჩევს ოთახებს, რომელთა ფასი მოცემულზე მეტია;

getRoomByPrice:: [(NedvObject,Int)] -> Int -> [(NedvObject,Int)]

getRoomByPrice [] \_ = []

getRoomByPrice ((House a,y):xs) price =

if y>price then (House a,y):getByPrice xs price

else getRoomByPrice xs price

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

if y>price then (Flat a b c,y): getRoomByPrice xs price

else getRoomByPrice xs price

getRoomByPrice ((Room a b c d,y):xs) price =

if y>price then (Room a b c d,y): getRoomByPrice xs price

else getRoomByPrice xs price

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

[(Room 4 120 9 20,1000)]

it :: [(NedvObject, Int)

-- getExceptBounds, ირჩევს მონაცემთა ბაზიდან ბინებს, რომლებიც არ მდებარეობს პირველ და ბოლო სართულებზე.

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

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