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

**ქულების განაწილება:** 1 - **5**ქ., 2 - **5** ქ., 3 - **5** ქ., 4 - **5** ქ., 5 - 1**5** ქ. =21 **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.
2. myMap :: (a -> b) -> [a] -> [b] +5
3. myMap f list = [(f x) | x <- list]
4. double x = x + x
5. myMap double [1, 2, 3, 4]
6. რეკურსიის საშუალებით. დაწერეთ მაღალი რიგის საბიბლიოთეკო myfilter ფუნქცია, რომელიც ირჩევს სიიდან ყოველ ელემენტს, რომელიც პრედიკატს აკმაყოფილებს განსაზღვრეთ ფუნქციის ტიპი. მოიყვანეთ გამოძახების მაგალითი.
7. myFilter :: (a -> Bool) -> [a] -> [a] +5
8. myFilter predicate [] = []
9. myFilter predicate (x:xs) | predicate x == True = concat [[x], (myFilter predicate xs)]
10. | otherwise = myFilter predicate xs
11. isPositive x = x >= 0
12. myFilter isPositive [(-1), 1, (-2), 2, 0]
13. 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.
14. analyzeList :: [Float] -> (Float, Float) +5
15. analyzeList [] = (0, 0)
16. analyzeList list = sumAndMult list 0 0 (length list)
17. sumAndMult :: [Float] -> Float -> Float -> Int -> (Float, Float)
18. sumAndMult [] sumSoFar multSoFar len = ((sumSoFar / len), (multSoFar \*\* (1 / len)))
19. sumAndMult (x:xs) sumSoFar multSoFar len = sumAndMult xs (sumSoFar + x) (multSoFar \* x) len
20. analyzeList [1, 2, 3]
21. განსაზღვრეთ maRali rigis *any* da *takeWhile* funqciebi standartul prelude failSi mocemuli maTi aRwerebis gamouyeneblad.
22. myAny :: (a -> Bool) -> [a] -> Bool +5
23. myAny predicate [] = False
24. myAny predicate (x:xs) = (predicate x) || (myAny predicate xs)
25. myAny isPositive [(-1), (-2), 2]

myTakeWhile :: (a -> Bool) -> [a] -> [a]

myTakeWhile predicate [] = []

myTakeWhile predicate (x:xs) | predicate x == True = concat[[x], (myTakeWhile predicate xs)]

                             | otherwise = []

myTakeWhile isPositive [1, 2, 3, (-1), 4]

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;
4. getByType - მეორე არგუმენტად გადაეცემა სტრიქონი, ფასი და ბაზიდან შესაბამის ინფორმაციას იღებს. მაგალითად,
5. **+1 magaliTi**

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)]