საგნის დასახელება: ფუნქციონალური პროგრამირება

სპეციალობა: კომპიუტერული მეცნიერება

ლექტორი: ნათელ არჩვაძე

დრო: 2 სთ

ქულათა ჯამი: 25=5+5+4+4+3

ქულათა გადანაწილების ინსტრუქცია: 5 საკითხი, თითოეული 5 ქულა.

ბილეთი=21

1. **განსაზღვრეთ შემდეგი ფუნქცია ორი ხერხით: პირობითი ოპერატორისა და დაცული განტოლებების გამოყენებით:  მნიშვნელობის გამოსათვლელად. როგორ მიმართავთ ამ ფუნქციას? ფუნქციის ტიპი განსაზღვრეთ ცხადად.**

let func x | x>=0=2^3-(7\*x)/3\*(2-x^2)| x<0&&x/=(-5)=(15-3\*x^2)/10+2\*x + 1/4\*(1-x) | x==(-5)=5\*(3+x/x^2+1)-225

Type -> func :: (Ord a, Fractional a) => a -> a

Call -> func 5 = 276.3333333333333

let func x = if x>=0 then 2^3-(7\*x)/3\*(2-x^2) else if x<0&&x/=(-5) then (15-3\*x^2)/10+2\*x + 1/4\*(1-x) else 5\*(3+x/x^2+1)-225

Type -> func :: (Ord a, Fractional a) => a -> a

Call -> func (-5) = -206.0

1. **განსაზღვრეთ შემდეგი ფუნქციები λ ნოტაციის გამოყენებით და მიუთითეთ შესაბამისი ტიპები:**

* ფუნქცია, რომელიც მოცემულ წყვილში ადგილებს უცვლის ელემენტებს.

let swap = (\(x, y)->(y,x))

Type -> swap :: (t1, t) -> (t, t1)

* ფუნქცია, რომელიც გადაცემული სიისთვის აბრუნებს 3-ის ჯერადი ელემენტების კვადრატებს, დანარჩენი ელემენტების კუბებს.

let func = \xs -> map (\x -> if mod x 3==0 then x^2 else x^3) xs

Type -> func :: [Integer] -> [Integer]

* ფუნქცია, რომელსაც გადაეცემა ორი სია და დააბრუნებს მათ გაერთიანებას.

let combine = \xs->(\cs->xs ++ cs)

Type->combine :: [a] -> [a] -> [a]

1. **განსაზღვრეთ ცხრილით მოცემული ფუნქცია რამდენიმე შაბლონის გამოყენებით:**

|  |  |  |
| --- | --- | --- |
| a | b | a⇒b |
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | False |

განსაზღვრეთ ფუნქციის ტიპი.

exor :: Bool -> Bool -> Bool

exor True True = False

exor False True = True

exor True False = True

exor False False = False

Type-> exor :: Bool -> Bool -> Bool სხვა შაბლ

1. **სიის კონსტრუქტორის გამოყენებით შეადგინეთ:**

* სამნიშნა რიცხვების სია, სადაც ყველა ციფრი განსხვავებულია;

[x|x<-[1,2,3]]

* გამოსახულება, რომლითაც დაითვლით 1000000–ზე ნაკლები 3–ის და 7–ის ჯერადი ნატურალური რიცხვების ნამრავლს და რაოდენობას?

sum [x | x <- [1..1000000], x `mod` 3 == 0 || x `mod` 7 == 0]

* სია [22, 32, ... 202 ]

[x^2|x<-[2..20]]

1. **განსაზღვრეთ ფუნქცია ორი ხერხით: REPL გარემოში და საწყისი კოდის სახით. ფუნქცია გამოიძახეთ კონკრეტული მონაცემებისთვის:**

* ფუნქცია, რომელიც აბრუნებს გაორმაგებულ არგუმენტს. -2

finddups :: Eq a => [a] -> [a]

finddups (x:xs) | x `elem` xs = x : finddups xs

| otherwise = finddups xs

finddups [] = []

* ფუნქცია max3, რომელიც სამი მთელი რიცხვიდან აბრუნებს მათ შორის უდიდესს.

let max3 a b c | a > b && a > c = a | b > a && b > c = b | c > a && c > b = c | otherwise = error "all numbers are the same" +2

max3 :: Integer -> Integer -> Integer -> Integer

max3 a b c

| a > b && a > c = a

| b > a && b > c = b

| c > a && c > b = c

| otherwise = error "all numbers are the same"

main = do

print(max3 1 2 3)

* ფუნქცია isRectangular, რომელიც პარამეტრად ღებულობს სიბრტყეზე სამი წერტილის კოორდინატებს და აბრუნებს True–ს, თუ მათ მიერ შედგენილი სამკუთხედი არის მართკუთხა სამკუთხედი.

isRectangular :: (Eq a, Floating a) => (a, a) -> (a, a) -> (a, a) -> Bool

isRectangular aXY bXY cXY = a \* (b \* b) == a \* (c \* c) + b \* (c \* c) ???? +1

where

a = calcDistance aXY bXY

b = calcDistance bXY cXY

c = calcDistance aXY cXY

calcDistance (x1, y1) (x2, y2) =

sqrt ((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1)))