საგნის დასახელება: ფუნქციონალური პროგრამირება

სპეციალობა: კომპიუტერული მეცნიერება

ლექტორი: ნათელ არჩვაძე

დრო: 2 სთ

ქულათა ჯამი: 30=5+5+5+0+4

ქულათა გადანაწილების ინსტრუქცია: 5 საკითხი, თითოეული 6 ქულა.

ბილეთი =19

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

**პასუხი:**

პირობითი ოპერატორის გამოყენებით:

func1 :: Float -> Float

func1 x = if x >= (-3) && x <= 7 then ((1-x^2+8.2\*x) / (x+2.5)) - 4

else if x == (-2.5) then 3.75 \* x^2 + (10 / ((2 - x)\*(1+4\*x)))

else 5\*x - 7 + ((3.5\*x^3) / (4 \* (1.5 + x)))

დაცული განტოლებების გამოყენებით:

func2 :: Float -> Float

func2 x | x >= (-3) && x <= 7 = ((1-x^2+8.2\*x) / (x+2.5)) - 4

| x == (-2.5) = 3.75 \* x^2 + (10 / ((2 - x)\*(1+4\*x)))

| otherwise = 5\*x - 7 + ((3.5\*x^3) / (4 \* (1.5 + x)))

ფუნქციას მივმართავთ:

func1 (-3)

func1 5

func2 10

func2 (-9)

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

* ფუნქცია, რომელიც აბრუნებს სიის მეორე ელემენტს.

**პასუხი:**

let second x = case x of (\_:x:\_) -> x

ტიპი: second :: [a] -> a

მაგალითი: second [1,5,7], აბრუნებს 5

* ფუნქცია, რომელიც გადაცემული სიისთვის აბრუნებს ლუწი ელემენტების კვადრატებს, კენტების -კუბებს.

**პასუხი:**

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

ტიპი: funcOddEven :: [Integer] -> [Integer]

მაგალითი: funcOddEven [1..9] აბრუნებს შედეგს: [1,4,27,16,125,36,343,64,729]

* ფუნქცია, რომელსაც გადაეცემა ელემენტი და სია და დააბრუნებს ახალ სიას, სადაც ბოლო ელემენტად ჩასმულია პირველი არგუმენტი.

**პასუხი:**

let funcInsertLast = \y -> \(xs) -> xs ++ [y]

ტიპი: fun :: a -> [a] -> [a]

მაგალითი: fun 2 [1,3] აბრუნებს შედეგს: [1,3,2]

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

|  |  |  |
| --- | --- | --- |
| a | b | a $$ b |
| False | False | True |
| False | True | True |
| True | False | False |
| True | True | True |

განსაზღვრეთ ფუნქციის ტიპი.

**პასუხი:**

let func3 a b = if a == b then True else if b == True then True else False

ტიპი: func3 :: eq a => a -> a -> Bool

შაბლონებით:

function False False = True

function False True = True

function True True = True

function \_ \_ = False

function :: Bool -> Bool -> Bool

1. სიის კონსტრუქტორის გამოყენებით შეადგინეთ:

იმ სამნიშნა რიცხვების სია, რომელთა ციფრთა ჯამი არის კენტი რიცხვი.

**პასუხი:**

let numbersList n=[x|x<-[1..n], `mod`x==1]

* შეადგინეთ ისეთი (x,y,z) სამეულების სია, სადაც თითოეული ცვლადი არის მარტივი, მნიშვნელობით 1–დან n-მდე და ჭეშმარიტია პირობა x+y==z.
* სია [22, 33, ... 2020].

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

* ფუნქცია, რომელიც აწყვილებს თავის ორ არგუმენტს.

**პასუხი:**

pairs [] = []

pairs xs = zip xs (tail xs)

* ფუნქცია, რომლის არგუმენტია სტრიქონი და რომელიც ამოწმებს, არის თუ არა არგუმენტი პალინდრომი.

**პასუხი:**

isPalindrome :: String -> Bool

isPalindrome xs = xs == reverse xs

ან

let isPalindrome = xs = xs == reverse xs

მაგალითი: isPalindrome “oto” აბრუნებს True, ხოლო isPalindrome “giorgi” კი აბრუნებს False

* ფუნქცია isSorted, რომელიც შესასვლელზე ღებულობს სამ რიცხვს და აბრუნებს True, თუ ეს რიცხვები დალაგებულია ზრდადობით ან კლებადობით.

**პასუხი:**

isSorted :: Integer -> Integer -> Integer -> Bool

isSorted x y z = if x < y then if y < z then True

else if x > y then if y > z then True

else False else False else False

ან

let isSorted x y z = if x < y then if y < z then True

else if x > y then if y > z then True

else False else False else False

მაგალითი: isSorted 1 5 7 აბრუნებს True, isSorted 7 5 1 აბრუნებს True, isSorted 9 3 2 აბრუნებს False