საგნის დასახელება: ფუნქციონალური პროგრამირება

სპეციალობა: კომპიუტერული მეცნიერება

ლექტორი: ნათელ არჩვაძე

დრო: 2 სთ

ქულათა ჯამი: 25=5+4+4+4+5

ქულათა გადანაწილების ინსტრუქცია: 5 საკითხი, თითოეული 5 ქულა.

მურად ჯანაშვილი

ბილეთი =22

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

1)დაცვით ოპერატორით :

let calc1 x | x==(-2.5)=3.75\*x^2+(10/((2-x)\*(1+4\*x))) | x>=(-3) && x<=7 = ((1-x^2+8.2\*x)/(x+2.5))-4 | otherwise=5\*x-7+3.5\*x^3/4\*(1.5+x)

calc1 :: (Ord a, Fractional a) => a -> a

გამოძახება: calc1 5

2)პირობითით

let calc x = if x == (-2.5) then 3.75\*x^2+(10/((2-x)\*(1+4\*x))) else if (x>(-3) && x<7) then ((1-x^2+8.2\*x)/(x+2.5))-4 else 5\*x-7+3.5\*x^3/4\*(1.5+x)

calc :: (Ord a, Fractional a) => a -> a

გამოძახება:calc 5

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

* ფუნქცია, რომელიც აბრუნებს სიის მეორე ელემენტს.

let secondElem = (\xs -> xs!!1)

secondElem :: [a] -> a

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

let oddEven = \xs -> map (\x -> if x `rem` 2==1 then x^3 else x^2)xs

oddEven :: [Integer] -> [Integer]

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

let pushToTail a = (\xs -> xs++[a]) -1

pushToTail :: a -> [a] -> [a]

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.

let ($$) False False =True

let ($$) False True = True

let ($$) True False = False

let ($$) True True = True

ტიპი : ($$) :: Bool -> Bool -> Bool

სხვა შაბლონი -1

1. სიის კონსტრუქტორის გამოყენებით შეადგინეთ:

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

პირველ რიგში ვსაზღვრავ რეკურსიულ ფუნქციას,რომელიც რიცხვებს დაშლის ციფრებად:

let digs :: Integral x => x->[x] ; digs 0 = [] ; digs x = x `mod` 10 : digs (x `div` 10)

შემდეგ კი ვიღებ [100,999] ინტერვალში მყოფ ყველა რიცხვს,ვშლი ჯერ-ჯერით ციფრებად,ვკრებ ამ ციფრებს და თუ ეს რიცხვი კენტია მაშინ ვამატებ სიაში :

[x | x<-[100..999] , (sum $ digs $ x) `rem` 2 == 1]

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

[x^x | x<-[1..20]]

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

* ფუნქცია, რომელიც აწყვილებს თავის ორ არგუმენტს.
* Wyvili a b = (a,b)  
  // repl let wyvili a b =(a,b)
* გამოძახება:

wyvili 1 2

(1,2)

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

isPalindrome str = str==reverse str

//repl let isPalindrome str = str==reverse str

გამოძახება:

isPalindrome "RacecaR"

True

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

isSorted a b c = ((a<b)&&(b<c)) || ((a>b)&&(b>c))

//repl let isSorted a b c = ((a<b)&&(b<c)) || ((a>b)&&(b>c))

გამოძახება:

isSorted 1 2 3

True

isSorted 8 5 2

True