საგნის დასახელება: ფუნქციონალური პროგრამირება

სპეციალობა: კომპიუტერული მეცნიერება

ლექტორი: ნათელ არჩვაძე

დრო: 2 სთ

ქულათა ჯამი: 25=5+5+5+5+5

ქულათა გადანაწილების ინსტრუქცია: 5 საკითხი, თითოეული 5 ქულა.

ბილეთი=25

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

let y x = if x>5 then (4\*x -5\*x^2 + 2)/(2+2x)-1/(3\*(x-1))

else if x<=5 && x/=(-8) then 3\*x^2 - 5/(2(x+8)) + 10

else (7\*x + 1/(1+x))/(x^2-1) +1/(2\*x)

Type : y :: (Ord a, Num (a -> a), Fractional a) => a -> a

let funY x |(x>5) = (4\*x -5\*x^2 + 2)/(2+2x)-1/(3\*(x-1))

|x<=5 && x/=(-8) = 3\*x^2 - 5/(2(x+8)) + 10

|otherwise = (7\*x + 1/(1+x))/(x^2-1) +1/(2\*x)

fun :: (Ord a, Num (a -> a), Fractional a) => a -> a

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

* ფუნქცია, რომელიც აბრუნებს გაორმაგებულ არგუმენტს.

პასუხი:

let fun = \x-> 2\*x

ფუნქციის ტიპი: fun :: Integer -> Integer

* ფუნქცია, რომელსაც გადაეცემა ორი არგუმენტი და ითვლის მთელ ხარისხს.

პასუხი: let fun = \x->(\y-> x^y)

ფუნქციის ტიპი: fun :: Integer -> Integer -> Integer

* ფუნქცია, რომელსაც გადაეცემა ორი წყვილი და დააბრუნებს შესაბამისი ელემენტების ჯამების წყვილს. მაგალითად, არგუმენტებია: (1,2), (3,4), შედეგია: (4,6).

პასუხი: let fun = \(a,b)->(\(c,d)->(a+c,b+d))

ფუნქციის ტიპი: fun ::

(Integer, Integer) -> (Integer, Integer) -> (Integer, Integer)

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

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

ფუნქცია დაწერილია საწყის ფაილში!

($$) :: Bool -> Bool -> Bool

True $$ False = False

\_ $$ \_ = True

ან უფრო სხვაგვარად :

($$) :: Bool -> Bool -> Bool

True $$ False = False

True $$ True = True

False $$ True = True

False $$ False = True

განსაზღვრეთ ფუნქციის ტიპი.

($$) :: Bool -> Bool -> Bool

1. სიის კონსტრუქტორის გამოყენებით შეადგინეთ:

* იმ სამნიშნა რიცხვების სია, რომელთა ციფრთა ჯამი არის 3-ის ჯერადი რიცხვი;

[x|x<-[100..999],(x 'mod' 10 + (x 'div' 10) 'mod' 10 + x 'div' 10) 'mod' 3==0]

* მოცემულ n რიცხვამდე ისეთი რიცხვების სია, რომელიც ორი მარტივი რიცხვის ჯამით; წარმოდგება.

let fun n = [x+y|x<-[1..n],y<-[x..n],length [a|a<-[1..x], mod x a==0]==2 && length [b|b<-[1..y], mod y b==0]==2]

* გამოსახულება , რომელიც პირველი 100 რიცხვის 12 + 22 + . . . + 1002 ჯამს ანგარიშობს.

sum [x^2|x<-[1..100]]

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

* ფუნქცია, რომელიც მოცემულ წყვილში ადგილებს უცვლის ელემენტებს.

REPL გარემოში:

let fun (a,b) = (b,a)

საწყისი კოდის სახით:

fun :: (a, b) -> (b, a)

fun (x,y) = (y,x)

ფუნქცია გამოიძახეთ კონკრეტული მონაცემებისთვის:

fun(10,50)

(50,10)

it :: (Integer, Integer)

* ფუნქცია, რომელიც გადაცემული რიცხვების სიისთვის ამოწმებს თითოეული ელემენტის ლუწ-კენტობას და გვიბრუნებს ლოგიკური მნიშვნელობების სიას.

REPL გარემოში:

let fun xs = map (\x->if (mod x 2)==0 then True else False) xs

საწყისი კოდის სახით:

fun :: [Integer] -> [Bool]

fun xs = map (\x->if (mod x 2)==0 then True else False) xs

ფუნქცია გამოიძახეთ კონკრეტული მონაცემებისთვის:

fun [1,2,3,4,5]

[False,True,False,True,False]

it :: [Bool]

* ფუნქცია isTriangle, რომელიც განსაზღვრავს, შეიძლება თუ არა მოცემულ x, y და z სიგრძის მონაკვეთებზე აიგოს სამკუთხედი.

REPL გარემოში:

let isTriangle x y z = (x+y>z)&&(x+z>y)&&(z+y>x)

საწყისი კოდის სახით:

isTriangle :: (Ord a, Num a) => a -> a -> a -> Bool

isTriangle x y z = (x+y>z)&&(x+z>y)&&(z+y>x)

ფუნქცია გამოიძახეთ კონკრეტული მონაცემებისთვის:

isTriangle 1 4 4

True

it :: Bool

isTriangle 2 3 5

False

it :: Bool