საგნის დასახელება: ფუნქციონალური პროგრამირება

სპეციალობა: კომპიუტერული მეცნიერება

ლექტორი: ნათელ არჩვაძე

ქულათა ჯამი: 15

ქულათა გადანაწილების ინსტრუქცია: 1 საკ -3ქ., 2 საკ -2ქ., 3 საკ -3ქ.,4 საკ-2 ქ., 5 საკ -5 ქ.

ბილეთი N=14

1. მოიყვანეთ არატრივიალური გამოსახულებების მაგალითები, რომლებიც ეკუთვნის ტიპებს: a)((Char,Integer), String, [Double])
2. [(([Double],[Bool]),[Integer])]
3. (Double,Bool,(String,Integer))

მოთხოვნა გამოსახულებების არატრივიალურობის შესახებ ნიშნავს, რომ გამოსახულებებში მონაწილე სიები უნდა შეიცავდნენ ერთ ელემენტზე მეტს.

ა)(('c',11),"ddd",[1.1,1.2,1.3,1.4])

ბ)[(([11.1,12.1,13.1],[True,False,False]),[11,12,13])] მეორე წევრი -1

გ)(12.3,True,("aaa",12))

1. როგორია შემდეგი მნიშვნელობების ტიპები:

a)([False,True],[’0’,’1’])

b)[’a’,’b’,’c’]

c)(“1”,”2”,”3”)

პას:

1. ([Bool], [Char])
2. [Char]
3. ([Char], [Char], [Char])

1. განსაზღვრეთ შემდეგი ფუნქციების მხოლოდ ტიპები პოლიმორფიზმის გამოყენებით:
2. pair x y = (x,y)
3. fun1 xs=(sum xs)\*(product xs)

c)length

d)[sum, product]

პას:

1. t -> t1 -> (t, t1)
2. Num a => [a] -> a
3. [a] -> Int
4. Num a => [[a] -> a]

თითოეული ფუნქცია გამოიძახეთ კონკრეტული მონაცემებისთვის.

a) pair 2 3

(2,3)

b) fun1 [1,2,3]

36

c) length [3,2,4,5,6]

5

d) [sum[1,2,3,4],product[1,2,3,4]]

[10,24]

1. მოცემულია ფუნქციის ტიპები:
2. fun :: Num a => a -> a
3. fun :: Ord a => (a, a) -> Bool

თითოეული ტიპისთვის მოიყვანეთ მაგალითი ფუნქციებისა, რომლებსაც ექნებათ მოცემული ტიპი.

პას:

1. fun x = x \* x
2. fun (a, b) = if a < b then True else False
3. მთელრიცხვა რიცხვითი კლასი

მთელრიცხვა რიცხვითი კლასი შეიცავს ტიპებს, რომელთა მნიშვნელობებიც მთელი რიცხვები და Num კლასის ეგზემპლარებია. კლასში გვხვდება Int და Integer ძირითადი ტიპები. int სასრული სიზუსტის მთელ რიცხვს წარმოადგენს.

ამ კლასში მხარდაჭერილია მრავალი მეთოდი, მათ შორის : +, - , \*, div და mod.

თუ ვმუშაობთ მთელი რიცხვების სიებთან, შეგვიძლია გამოვიყენოთ სტანდარტული ფუნქციები (length, head, take, drop) და ა.შ.

მაგალითად :

**\*Test>** 3 `div` 4

0

it :: Integer

**\*Test>** 3 `div` 2

1

it :: Integer

**\*Test>** 8 `mod` 3

2

it :: Integer

**\*Test>** 9 `mod` 2

1

it :: Integer

**\*Test>** 3 + 5

8

it :: Integer

**\*Test>** head [1,2,3,4]

1

it :: Integer

**\*Test>** tail [2,3,4,4]

[3,4,4]

it :: [Integer]

**\*Test>** take 1 [1,2,3,4]

[1]

it :: [Integer]

**\*Test>** take 2 [1,2,3,4]

[1,2]