Chetan Hiremath

Dr. Johnson

EECS 368

April 11, 2022

EECS 368 Assignment 6

Haskell Source Code:

--Replicate Function:

--This line defines the replicate function. It considers Int and x as the inputs and [x] as the output, which is enclosed with brackets since you need the

--identical elements in a list.

replicate':: Int -> x -> [x]

--This line calculates and prints the list of the identical elements when the element is cloned for the fixed number of times.

replicate' x xs = [xs | \_ <- [1..x]]

--Perfects Function:

--This line defines the factors function since the factors function is mandatory for the perfects function. It considers Int as the input and [Int] as the output.

--[Int] is enclosed with brackets since you need the list of factors of the values.

factors :: Int -> [Int]

--This line calculates and prints the lists of values' factors.

factors n = [x | x <- [1..n], n `mod` x == 0]

--This line defines the perfects function. It considers Int as the input and [Int] as the output.

--[Int] is enclosed with brackets since you need the list of perfect values.

perfects :: Int -> [Int]

--This line calculates and prints the list of perfect values. It uses the init functon to remove the last elements of the lists.

--Then, the sum function will returns the sums of the lists' respective factors. If the sum of the factors of the list is equal

-- to the value of the factors, then it is a perfect value. Then, the perfect values are added in the list that is printed by this function.

perfects n = [x | x <- [1..n], sum (init (factors x)) == x]

--Positions Function:

--This line defines the find function since the find function is mandatory for the positions function. It considers a and [(a,a)] as the inputs and [a] as the output.

--[(a,a)] and [a] are enclosed with brackets since you need the lists to find the positions of the same values. Eq means equality since it checks the equality.

find :: Eq a => a -> [(a,a)] -> [a]

--This line calculates and prints the list of the positions of the same values. If the value is equal to the position's value, then its position is added in the list.

find x xs = [p | (x', p) <- xs, x == x']

--This line defines the positions function. It considers Int and [Int] as the inputs and [Int] as the output.

--[Int] is enclosed with brackets since you need the list of the positions.

positions :: Int -> [Int] -> [Int]

--This line calculates and prints the list of the positions of the same values. The zip function maps 2 lists to a list of pairs of their corresponding elements.

--Then, the find function checks the positions of the same values and sends the list to the positions function that prints that list.

positions p xs = find p (zip xs [0..((length xs) - 1)])

--Scalar Product Function:

--This line defines the scalarproduct function. It considers [Int] and [Int] as the inputs and Int as the output.

--[Int] and [Int] are enclosed with brackets since you need 2 lists to calculate the scalar product, which is one the most important formulas of Calculus 2 and 3.

scalarproduct :: [Int] -> [Int] -> Int

--This line calculates and prints the scalar product of 2 lists. The zip function maps 2 lists to a list of pairs of their corresponding elements.

--Then, the x and the y values of the respective pairs are multiplied. Those pairs are added by the sum function that will print the final value of the scalar product.

scalarproduct xs ys = sum [x \* y | (x,y) <- zip xs ys]

Haskell Outputs:

Replicate Function:

> replicate 3 Test

[Test,Test,Test]

> replicate 5 "test code"

["test code","test code","test code","test code","test code"]

Perfects Function:

> perfects 500

[6,28,496]

> perfects 9000

[6,28,496,8128]

Positions Function:

> positions 0 [1,0,0,1,0,1,1,0]

[1,2,4,7]

> positions 1 [1,0,0,1,0,1,1,0]

[0,3,5,6]

Scalar Product Function:

> scalarproduct [1,2,3][4,5,6]

32

> scalarproduct [-1,2,3][-4,-5,6]

12