HASKELL TUTORIAL 2

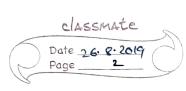
Prelude = inselet :: [Float] > [Float] > Float melet la lb = n= len la diff = zipwith (-) lalb diffsq = map (2) diff (foldr (+) 0 diff sq.) /n meWhere: [Float] -> [Float] -> Eloat msewhere la 16 = (foldr (+) 0 diffsq)/n n = len la diff = zipwith (-) la lb diff Sq = map (12) diff

il / :load

: + /: type - typeoof object.

type Vector = [Float]

cimilar to typedef.

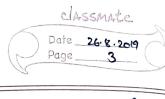


type Fe' = Float k:: FL' -> FL' -> FL' type vector = [Float] data Tree = Empty | Node Int Tree Tree 6: Tree -> String 6 Empty = "empty node" (Node a b c) = "full node" b : Tree > Int 6 Empty = -1 f (Node a - _) = a data Vector' = Vector' Int [Float] data Vector" = Vector" & aim ! Int, vector ! [Float] } get Dims : Vector -> In+ data Tree = Empty | Mode Int Tree Tree

b: Tree -> Bool

b = mpty = true

b - = false.



	data Vector" = Vector" Saim: Int, vector : [Float]}
\/	class show a coher show it a -> string
X	Show : 1 a -> String
	chetance Show Vector 11 where
	show v = (show & dim a) ++ "++ (show & vector v)
	The state of the s

× hello world.