Exercise 12

*-- Define Map*

*-- type of treeMap :: (a -> b) -> Tree a -> Tree b*

treeMap :: (a -> b) -> Tree a -> Tree b

treeMap f Empty = Empty

treeMap f (Node l v r) =

Node (treeMap f l) (f v) (treeMap f r)

*-- Map with fold*

*-- type of treeFoldr :: (a -> b -> b) -> b -> Tree a -> b*

treeFoldr f acc Empty = acc

treeFoldr f acc (Node l v r) =

let acc' = treeFoldr f acc l

acc'' = treeFoldr f acc' r

in f v acc''

*-- find height*

height Empty = 0

height (Node l \_ r) =

max (height l) (height r) + 1

*-- define isBST (copy from the slide)*

*-- type of isBST :: Ord a => Tree a -> Bool*

isBST t = fst .

foldl lt (True, Nothing) $

inorder t

where

lt (False, \_) \_ = (False, Nothing)

lt (True, Nothing) x = (True, Just x)

lt (True, Just b) x = (b <= x, Just x)