Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Функциональное и логическое программирование»

Отчет к лабораторной работе № 7

«Удаление из бинарного дерева на haskell»

Выполнил:

Василения Иван Валерьевич

Мельниченко Артём Олегович

Принял:

Гордиенко А.П.

Орёл, 2024г

module Main where

data Tree a = Empty | Node a (Tree a) (Tree a)

insert :: Ord a => a -> Tree a -> Tree a

insert x Empty = Node x Empty Empty

insert x (Node y left right)

| x >= y = Node y left (insert x right)

| otherwise = Node y (insert x left) right

delete :: Ord a => a -> Tree a -> Tree a

delete x Empty = Empty

delete x (Node y left right)

| x < y = Node y (delete x left) right

| x > y = Node y left (delete x right)

| otherwise = case (left, right) of

(Empty, \_) -> right

(\_, Empty) -> left

(\_, \_) -> let minVal = findMin right

in Node minVal left (delete minVal right)

findMin :: Tree a -> a

findMin (Node x Empty \_) = x

findMin (Node \_ left \_) = findMin left

takeTreeAsList :: Tree a -> [a]

takeTreeAsList Empty = []

takeTreeAsList (Node x left right) =

takeTreeAsList left ++ [x] ++ takeTreeAsList right

main :: IO ()

main = do

let tree = foldr insert Empty [10, 5, 15, 7, 12]

let newTree = delete 7 tree

print (takeTreeAsList tree)

print (takeTreeAsList newTree)