Lab 6: Binary Search Tree

In this lab, we will implement binary search tree ADT with the three main functions, insert, delete, and find. Additionally, we will have three print functions with different ways of traversal.

* Input

Obtain a list of operations from the given input file, and execute the given operations in order. A detailed specification of the operations is provided below. Each line represents a single operation. Each operation and the necessary parameters are separated by a space. You may assume that the input values (represented as x below) are any integer.

* **i x**: insert a new key “x” into the binary search tree without duplication. If x already exists in the tree, print an error message.
* **d x**: delete a key “x” in the binary search tree. If x does not exist in the tree, print an error message.
* **f x**: find the given key to check whether the key exists in the tree
* **pi**: print the tree by inorder traversal
* **pr:** print the tree by preorder traversal
* **po:** print the tree by postorder traversal

An input file is shown below.



|  |
| --- |
| Macintosh HD:Users:minarho:Downloads:output_lab6.png |

* Binary Search Tree ADT

(1) Data Specification for the objects

struct Tree {

int value;

Tree \*left;

Tree \*right;

};

(2) Function specification

- Tree\* insertNode(Tree \*root, int key)

-Insert a new node with the key value into the tree. If the key already exists in the tree, print an error message.

- Tree\* deleteNode(Tree \*root, int key)

-delete a node with the given key value from the tree. If the key does not exist in the tree, print an error message.

- Tree\* findNode(Tree \*root, int key)

- Find the key in the binary search tree.

- Print “key is in the tree” if the key exists. Otherwise, print “key is not in the tree”.

- void printInorder(Tree \*root)

- Print the tree by inorder traversal.

- void printPreorder(Tree \*root)

- Print the tree by preorder traversal.

∙void printPostorder(Tree \*root)

- Print the tree by postorder traversal.

(3) Program description

* name : p6.c
* input : a list of operations in a file (an input file name is given as a command line argument. See the example in “input” on the first page)
* output : the corresponding result in the standard output

Submit to the course gitlab your source code. (~2020/4/30 23:59)