January 2023 CSE 106
Offline Assignment on Binary Search Tree

Deadline: Saturday, 22 July 2023, 11:55 PM
Last modified : Tuesday 18th July, 2023 20:21

Current Version	Time of Modification	Modification
1.1	Tuesday 18th July, 2023 13:57	Usage of other data structures (array, stack, queue) is strictly prohibited.

A **Binary Search Tree** (BST) is a binary tree data structure which has the following properties:

1. The left subtree (if exists) of a node contains only nodes with keys lesser than the node's key.
2. The right subtree (if exists) of a node contains only nodes with keys **greater** than the node's key.
3. The left and right subtrees each must also be a binary search tree.

We assume that all the keys will be **unique**.

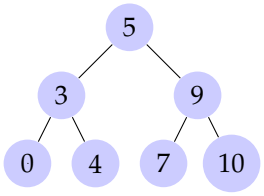

Figure 1: A sample BST

For this assignment, you will need to implement a Binary Search Tree yourself. Our designed BST has the following operation,

1. **Insert** It inserts a key in the BST.
2. **Delete** It deletes the key in the BST, if it exists in the BST. Remember, deletion in the BST can be divided into three cases.
   (a) **Node to be deleted is the leaf:** Simply remove it from the tree.
   (b) **Node to be deleted only has one child:** Copy the child to the node and delete the child
   (c) **Node to be deleted has two children:** Find the inorder successor of the node. Copy contents of the inorder **successor** to the node and delete the inorder successor.
3. **Find :** Searches whether a key is present in our BST.
4. **Traversal:** Traverses the tree as specified. Traversal can be of three types.
   (a) In-order
   (b) Post-order
   (c) Post-order

Please notice that you have to implement these functions with the optimal possible solution. Roughly all operations (except traversal) should take $O(logn)$ time. This means if currently you have n keys in the BST, your operation (insert/delete/find) should take around $log(n)$ comparison on average.

Input

Take input from a file named *in.txt*.
You will generate output on a file named *out.txt*
Each line in the input will specify one of the following operations:

1. Insert (I) followed by an integer denoting the value of the node to be inserted
2. Delete (D) followed by an integer denoting the value of the node to be deleted
3. Find (F) followed by an integer denoting the value of the node to be searched for
4. Traversal (T) followed by the type of traversal: In-order (in), Pre-order (pre), or Post-order (post)

Output

After,

1. Insert: the current state of the tree after insertion will be printed in the nested parentheses format. For example, the nested parentheses format for the tree in Figure 1 will be written as:

```
(5(3(0,4),9(7,10)))
```

2. Delete: same as insert
3. Find: print "found" or "not found"
4. Traversal: print the keys in specified mode separated by space. An in-order traversal in 1 would be:

```
0 3 4 5 7 9 10
```

Please refer to the sample test cases for a better understanding of input-output format. You may use this website for comparing your output with the sample one. Please remember that your offline would be checked using automated scripts. **Failure to follow the output format will result in a heavy mark deduction.**

Sample I/O

Input File

```
F 1
I 8
I 10
I 3
I 1
I 14
I 6
I 4
I 13
I 7
T In
T Pre
T Post
D 8
D 7
D 10
D 10
F 4
```

Output

```
not found
(8)
(8(,10))
(8(3,10))
(8(3(1,),10))
(8(3(1,),10(,14)))
(8(3(1,6),10(,14)))
(8(3(1,6(4,)),10(,14)))
(8(3(1,6(4,)),10(,14(13,))))
(8(3(1,6(4,7)),10(,14(13,))))
1 3 4 6 7 8 10 13 14
8 3 1 6 4 7 10 14 13
1 4 7 6 3 13 14 10 8
(10(3(1,6(4,7)),14(13,)))
(10(3(1,6(4,)),14(13,)))
(13(3(1,6(4,)),14))
(13(3(1,6(4,)),14))
found
```

Submission

1. Create a directory with your 7-digit student ID as its name.
2. Put all the source files (.cpp/.hpp/.c/.h files) only into the directory created in step 1.
3. Zip the directory (compress in .zip format. Any other format like .rar, .7z etc. are not acceptable).
4. Upload the .zip file in moodle.

Mark Distribution

	Task Detail	Marks
1	Insert Function	20
2	Proper Implementation of Delete Function	40
3	Find Function	10
4	Traversal Functions	10
5	Proper Formatting	10
6	Proper Submission	10

For Queries

If you have any questions related to the assignment please first check the queries thread in moodle. You should post your confusion in the thread. If your query goes unanswered for 24 hours, please mail me.

Special Instructions

When writing code, it is essential to ensure readability, reusability, and good structure. This involves using appropriate functions to implement algorithms, giving variables meaningful names, adding suitable comments when necessary, and maintaining proper indentation. You will need to use your offline implementation to solve the onlines. There will be a viva too. So, please understand the concepts before you proceed to code.

Please note that you must rely on your own implementation to solve the assigned tasks. It is strictly prohibited to copy code from any source, including friends, seniors, or the internet. **Any form of plagiarism, regardless of its origin or destination, will result in a deduction of 100% marks for the offline assessment**. Moreover, repeated instances of plagiarism may lead to stricter consequences in accordance with departmental policies.