Offline - 3

Binary Search Tree

Deadline: 16th December 11:59 pm

Prepared by: Syed Muhammad

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

- The left subtree of a node contains only nodes with keys **lesser** than the node's key.
- The right subtree of a node contains only nodes with keys **greater** than the node's key.
- The left and right subtrees each must also be a binary search tree.

For this assignment, you will get to implement a Binary Search Tree yourself. The input to your code will be read from a text file named "`input.txt`" and you will generate the output on a text file named "`output.txt`". Each line in the input will specify one of the following operations:

- **Insert** (`I`) followed by an integer denoting the value of the node to be inserted
  Returns: the current state of the tree after the insertion*

- **Delete** (`D`) followed by an integer denoting the value of the node to be deleted
  Returns: the current state of the tree after the deletion*

- **Find** (`F`) followed by an integer denoting the value of the node to be searched for
  Returns: True or False depending on the success of the operation

- **Traversal** (`T`) followed by the type of traversal:
  In-order (`In`), **Pre-order** (`Pre`), or **Post-order** (`Post`)
  Returns: A linear ordering of each node in the tree traversed as specified

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

```
10(6(4)(8(7)(9)))(11()(17()(21)))
```

You can assume **no duplicate nodes** will be placed as input. And deletion operations can only be done on **leaf nodes**. Attempting to delete any other node will output "`Invalid Operation`".

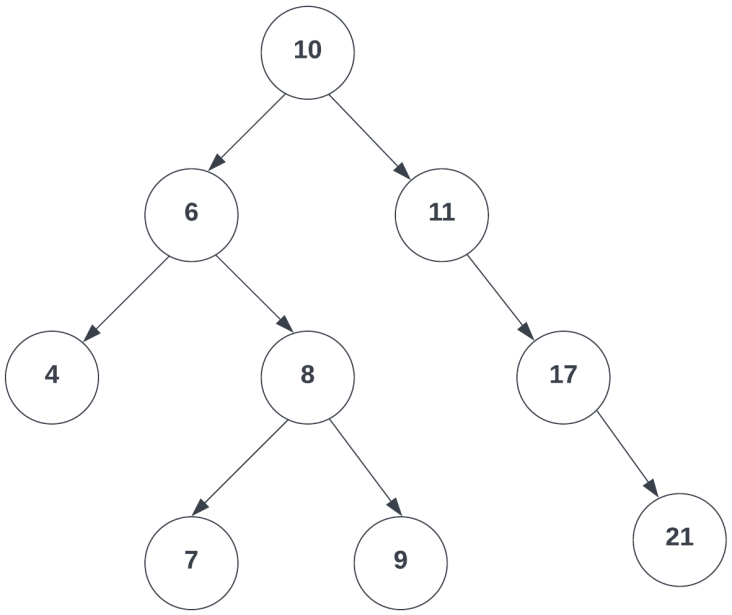

Figure : 1

Sample I/0:

Input	Output
I 10 I 11 F 6 I 6 F 6 I 4 F 11 I 17 I 8 I 7 I 21 I 9 T In T Pre T Post F 9 D 9 F 9 D 11 F 11	10 10()(11) False 10(6)(11) True 10(6(4)())(11) True 10(6(4)())(11()(17)) 10(6(4)(8))(11()(17)) 10(6(4)(8(7)()))(11()(17)) 10(6(4)(8(7)()))(11()(17()(21))) 10(6(4)(8(7)(9)))(11()(17()(21))) 4 6 7 8 9 10 11 17 21 10 6 4 8 7 9 11 17 21 4 7 9 8 6 21 17 11 10 True 10(6(4)(8(7)()))(11()(17()(21))) False Invalid Operation True

Sample “input.txt”:

```
I 10
I 11
F 6
I 6
F 6
I 4
F 11
I 17
I 8
I 7
I 21
I 9
T In
T Pre
T Post
F 9
D 9
F 9
D 11
F 11
```

Sample “output.txt”:

```
10
10()(11)
False
10(6)(11)
True
10(6(4)())(11)
True
10(6(4)())(11()(17))
10(6(4)(8))(11()(17))
10(6(4)(8(7)()))(11()(17))
10(6(4)(8(7)()))(11()(17()(21)))
10(6(4)(8(7)(9)))(11()(17()(21)))
4 6 7 8 9 10 11 17 21
10 6 4 8 7 9 11 17 21
4 7 9 8 6 21 17 11 10
True
10(6(4)(8(7)()))(11()(17()(21)))
False
Invalid Operation
True
```

House-Keeping:

While you are encouraged to talk to your peers, ask help from teachers, and search relevant resources online, please do not copy your code from any source.
Reiterating that the penalty for plagiarism is a negative 100%