Lab 8 Binary search tree & tree traversal

Input will be n integer numbers, and you should construct these numbers into a binary search tree using linked **list**. You can assume that the value of each element will not repeat.

There are three main functions to be completed in BST.cpp .

- `void Insert(Node* node)` : Insert the node to the binary search tree.
- `void Inorder_traversal(Node* root)` : Use in-order method to traverse binary search tree.
- `void Level_traversal(Node* root)` : Use level order method to traverse binary search tree.

Notice that:

1. You must complete Node.cpp before you construct the binary search tree.
2. The value of each node must be greater than any value stored in the left sub-tree, and less than any value stored in the right subtree.
3. You must use linked list to construct the binary search tree.

Input Format

Read the input data from "input1.txt".

The first line shows the number of test cases.
Every case includes two lines.
The first line is an integer which represents the number of nodes to add to the binary tree.
In the second line, each number represents the value of a node.

Output Format

For every binary search tree, print the result of in-order and level order traversal.
See more detail from sample output.

Sample Input(input1.txt)

```
2

10
9 20 6 17 38 92 41 27 56 3

12
6 15 13 2 50 81 35 1 14 5 12 37
```

Sample Output

```
Inorder_traversal: 3 6 9 17 20 27 38 41 56 92
Level_traversal: 9 6 20 3 17 38 27 92 41 56

Inorder_traversal: 1 2 5 6 12 13 14 15 35 37 50 81
Level_traversal: 6 2 15 1 5 13 50 12 14 35 81 37
```

Hint

In-Order Traversal:
You can use this traversal to print an sorted order in a binary search tree.

```
// InOrder traversal algorithm
inOrder(TreeNode<T> n) {
    if (n != null) {
        inOrder(n.getLeft());
        visit(n)
        inOrder(n.getRight());
    }
}
```