

**Question - 1**
BST**SCORE: 5 points**

In delete operation of BST, we need inorder successor (or predecessor) of a node when the node to be deleted has both left and right child as non-empty. Which of the following is true about inorder successor needed in delete operation?

- ☐ Inorder Successor is always a leaf node
- ☒ Inorder successor is always an empty left child
- ☐ Inorder successor may be an ancestor of the node
- ☐

Inorder successor is always either a leaf node or a node with empty right child

Question - 2
BST2**SCORE: 5 points**

We are given a ordered set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given ordered set so that it becomes a binary search tree?

- ☐ 0
- ☒ 1
- ☐ $n!$
- ☐ $(1/(n+1)) \cdot 2n C n$

Question - 3
BST3**SCORE: 5 points**

Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?

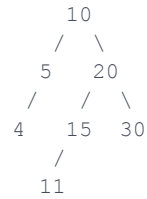
- ☐ 7 5 1 0 3 2 4 6 8 9
- ☐ 0 2 4 3 1 6 5 9 8 7
- ☒ 0 1 2 3 4 5 6 7 8 9

Question - 4

BST4

SCORE: 5 points

Consider the following Binary Search Tree:



If we randomly search one of the keys present in above BST, what would be the expected number of comparisons?

- ☐ 2.75
- ☐ 2.63
- ☒ 2.57
- ☐ 3.25

Question - 5

BST5

SCORE: 30 points

Implement binary search tree CountNodes and Search operations.

The first line of the input is number of operations that will happen.
The subsequent lines represent the operation and key, value pair.

Sample Input1:

```

4 // number of operations
1,5 //(operation,nodeValue)
1,1
1,3
2,3

```

Sample Output1:

```

true

```

Sample Input2:

```

4 // number of operations
1,5 //(operation,nodeValue)
1,1
1,3
4

```

Sample Output2:

```

3

```

Types of Operation:

```
1 -> insert (inserts an element into the tree)
2 -> search (returns true/false depending on
whether or not the value is found)
3 -> isEmpty (returns true for an empty tree else
false)
4 -> countNodes (returns number of nodes in the
tree)
5 -> preorder traversal
6 -> inorder traversal
7 -> postorder traversal
```