**UCS 2312 Data Structures Lab**

**Assignment 5: BSTADT and its application**

Create an ADT for the binary search tree data structure with the following functions. Each node which consists of integer data, address of left and right children.

[CO2, K3]

1. insertBST(t,data) – insert data into BST
2. inorder(t) – display the tree using inorder traversal
3. preorder(t) – display the tree using preorder traversal
4. postorder(t) – display the tree using postorder traversal
5. levelorder(t) – display the tree hierarchically
6. findmin(t)– returns the minimum element in the tree
7. search(t,key) – returns the element found, otherwise returns NULL
8. delete(t,elt) – delete the given elt from tree

1. Demonstrate the BSTADT with the following test

case

Insert(t,29)

Insert(t,23)

Insert(t,4)

Insert(t,13)

Insert(t,39)

Insert(t,31)

Insert(t,45)

Insert(t,56)

Insert(t,49)

Inorder(t) → 4,13,23,29,31,39,45,49,56 Levelorder(t)→ 1st Level → 29

2nd level → 23, 39

3rd Level →4, 31, 45

4th Level →13, 56

5th Level → 49 Findmin(t) → 4

Find(t, 13) → Found, value is

3 Find(t,3) → Not found

2. Write an application to do the following

a. Given a sorted array. Write a function that creates a Balanced Binary Search Tree using array elements.

Examples:

Input: arr[] = {1, 2, 3}

Output: A Balanced BST

2

/ \

1 3

Explanation: all elements less than 2 are on the left side of 2 , and all the elements greater than 2 are on the right side

Input: arr[] = {1, 2, 3, 4}

Output: A Balanced BST

3

/ \ 2 4

/

1

b. Given a Binary Search Tree (BST), find the second largest element.

**Example:**

***Input:*** *Root of below BST*

*10*

*/*

*5*

# Output: 5

***Input:*** *Root of below BST*

*10*

*/ \*

*5 20*

*\*

*30*

# Output: 20

1. Count the number of nodes in tree within the given range
2. Find sum of k smallest elements in the given BST



Test case for the

Application (a)

Input: Tree 1

10

/ \

5 50

/ / \

1 40 100 Input: Tree2

10

/ \

5 50

/ / \

1 40 100

Tree1 and Tree2 are identical with a set of elements (b) Tree1 not complete (c) Tree1 Range: [**5**, 45] Output: 3

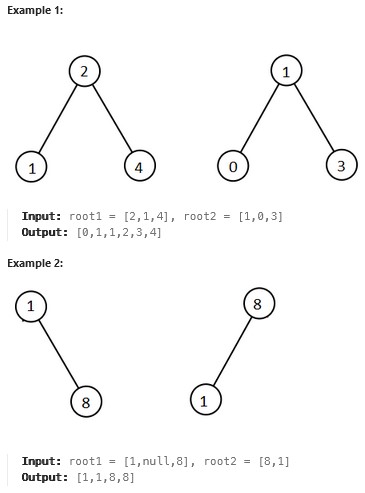
Nodes are 5, 10, 40

Tree2 Range: [**1**, 45]

Output: 4

Nodes are 1,5, 10, 40

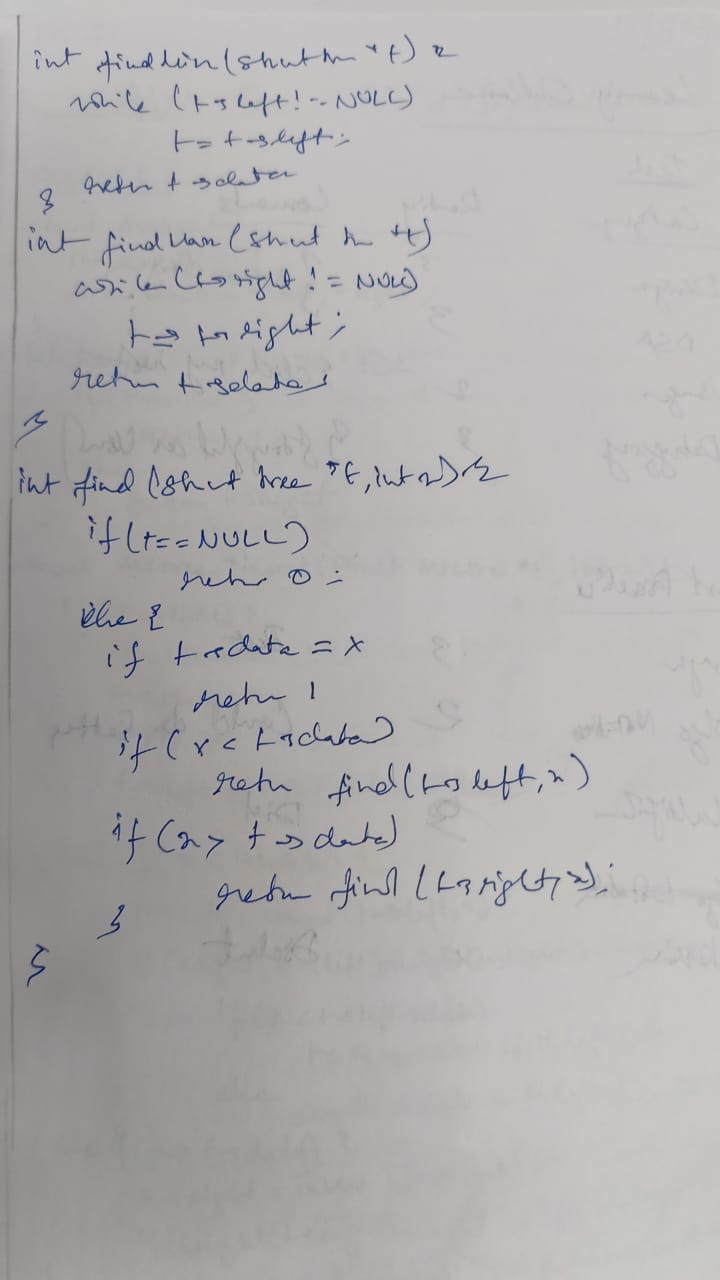
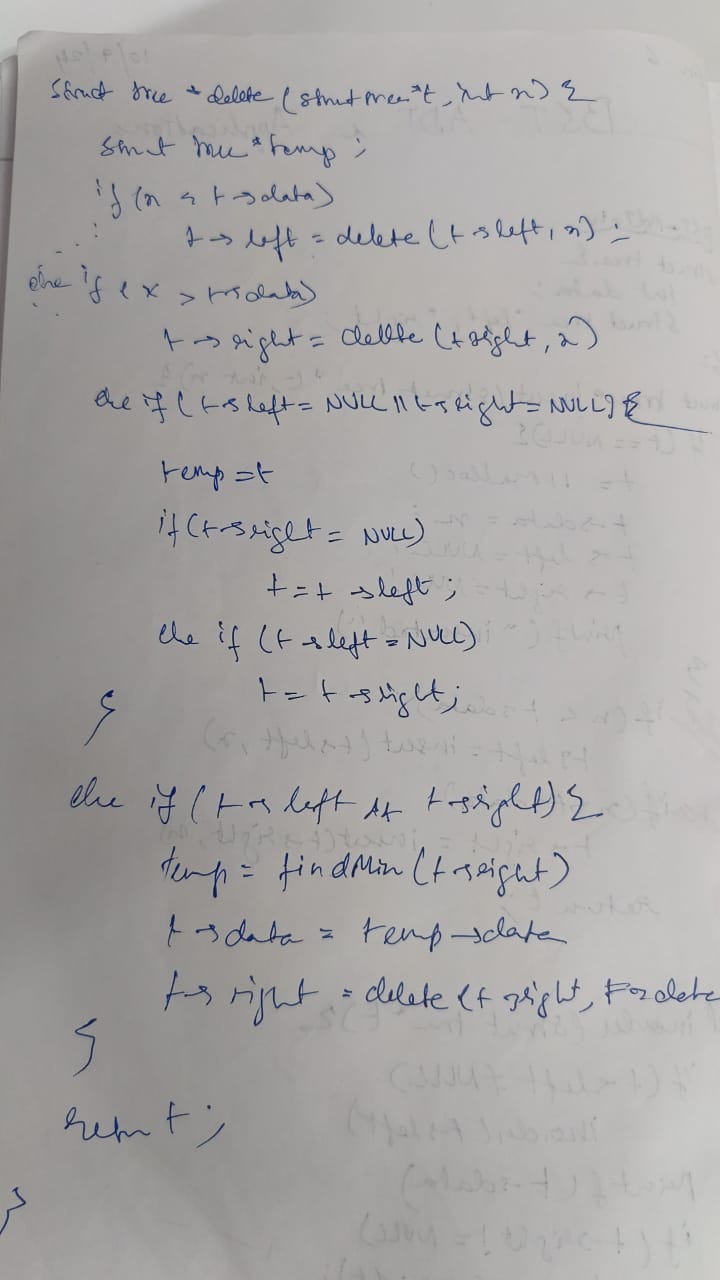
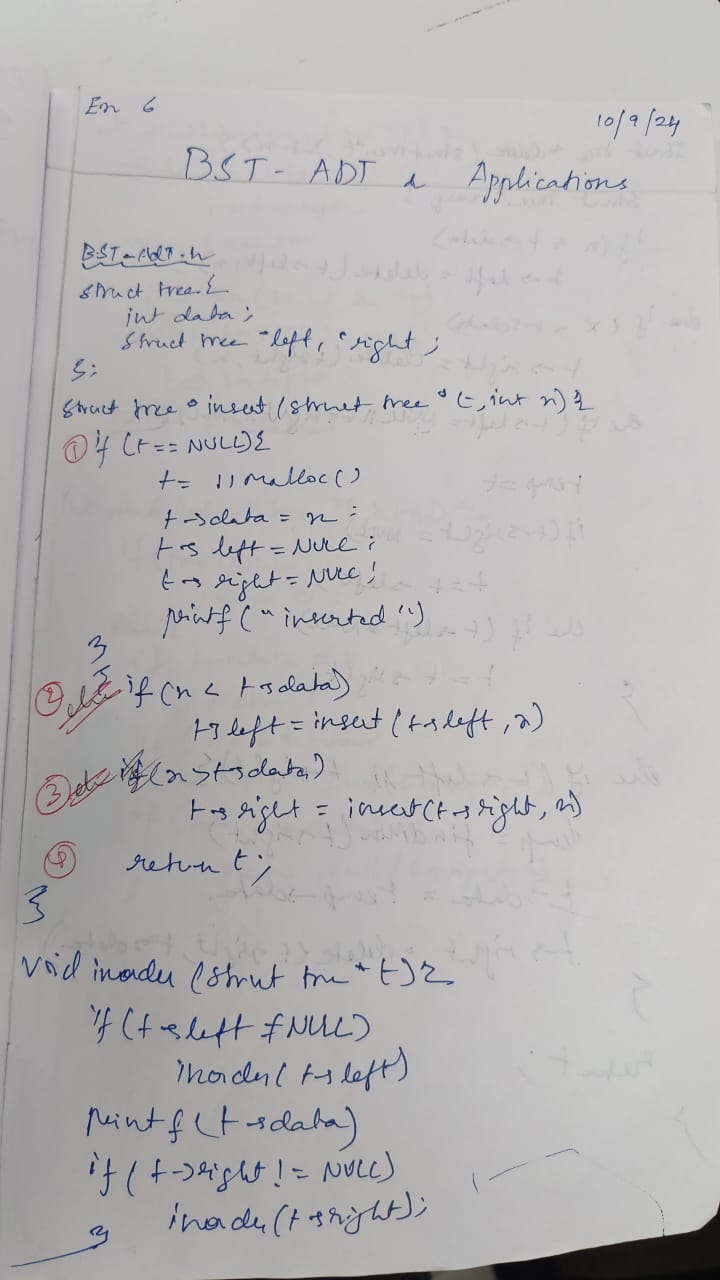
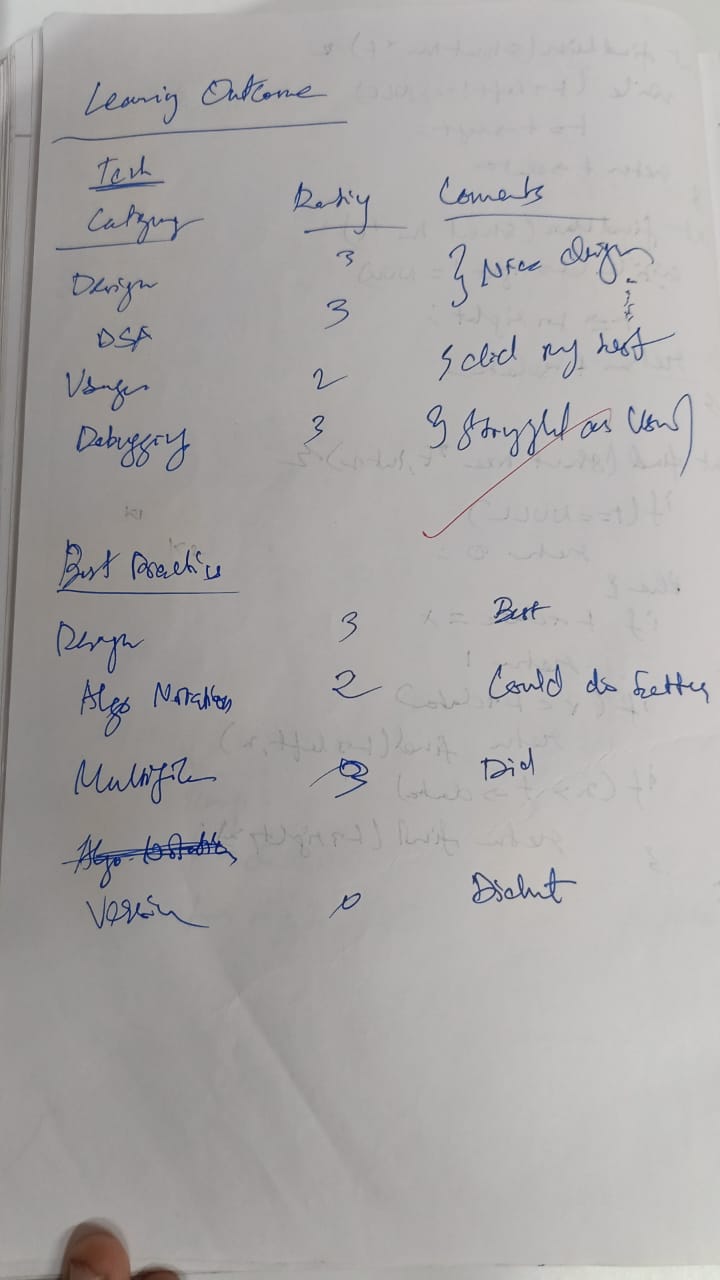
Given two binary search trees root1 and root2, return *a list containing all the integers from both trees sorted in* **ascending** *order*.

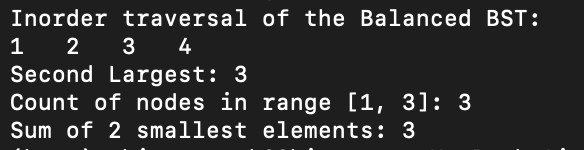
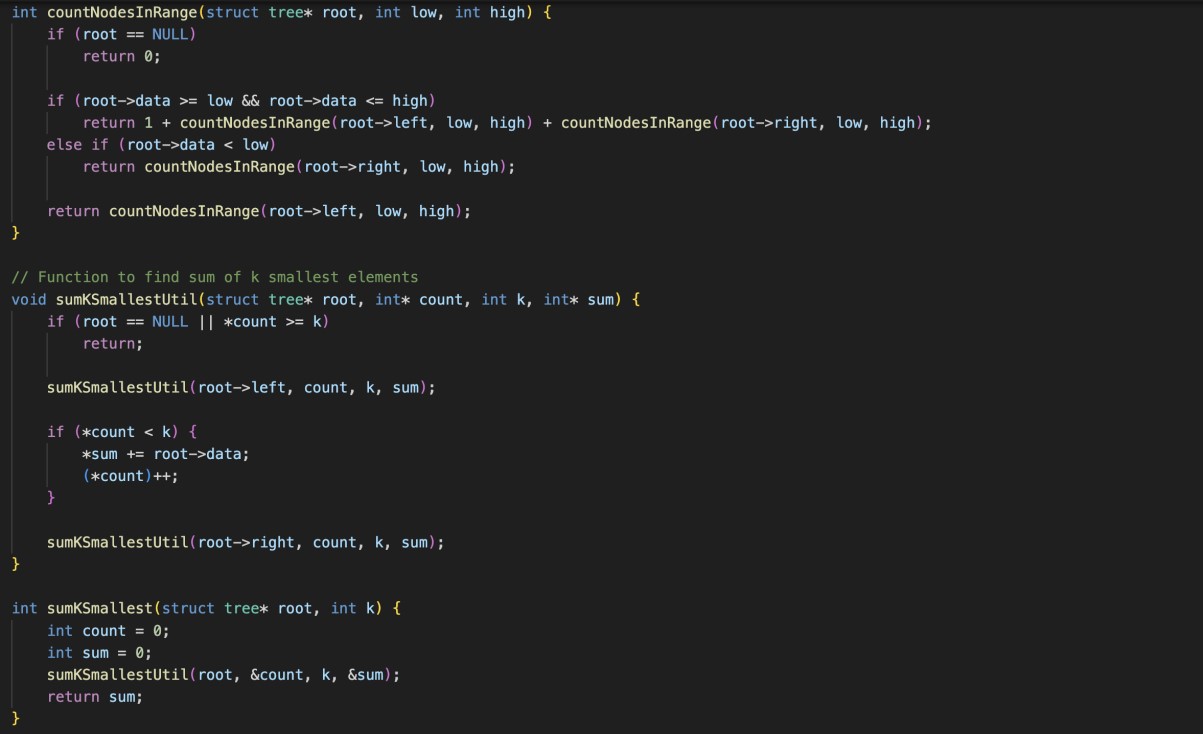
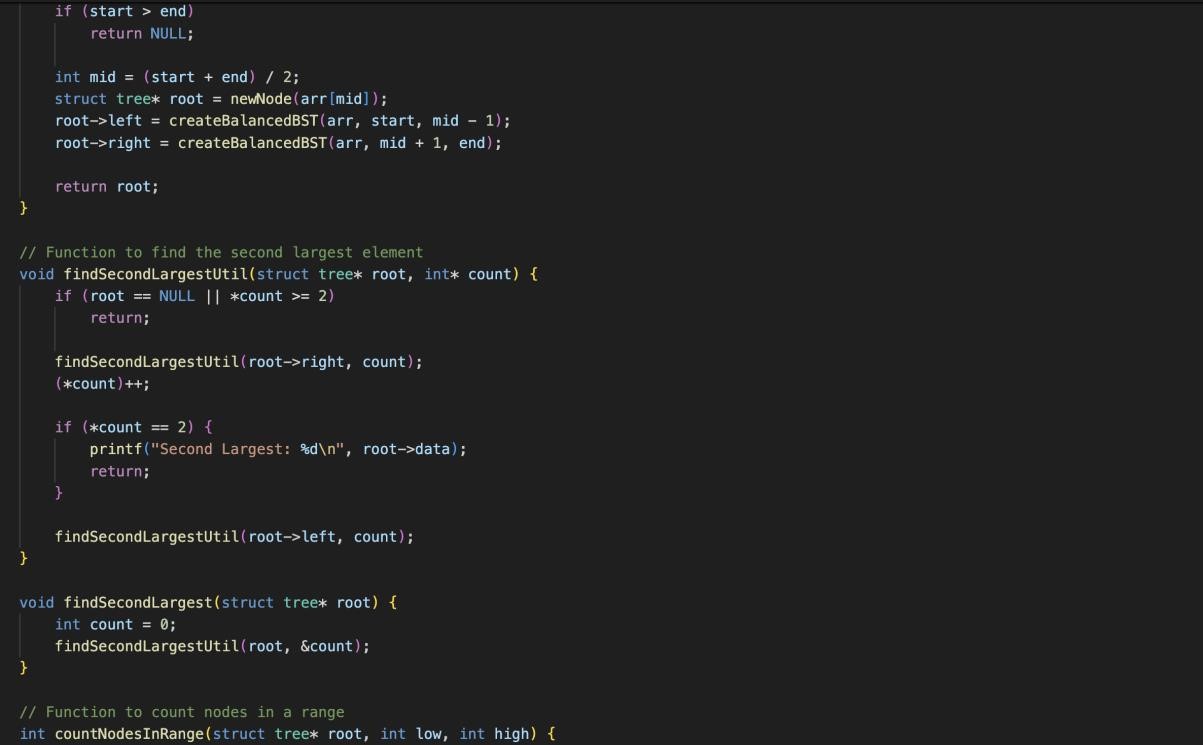
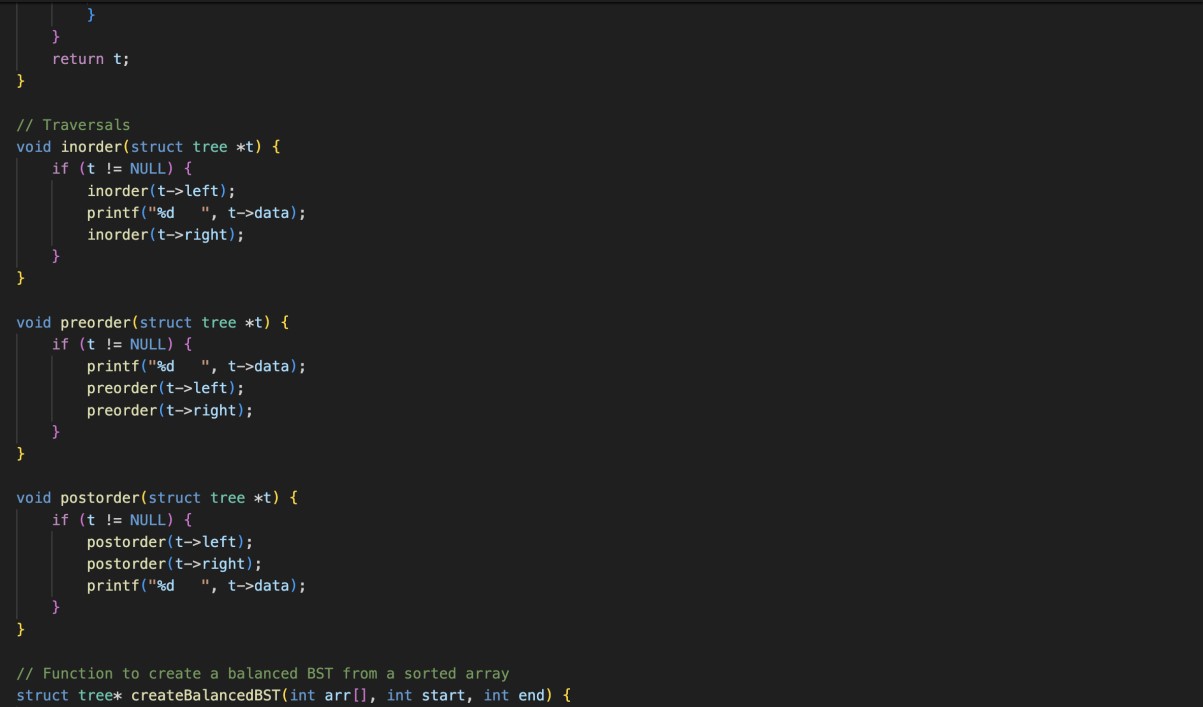
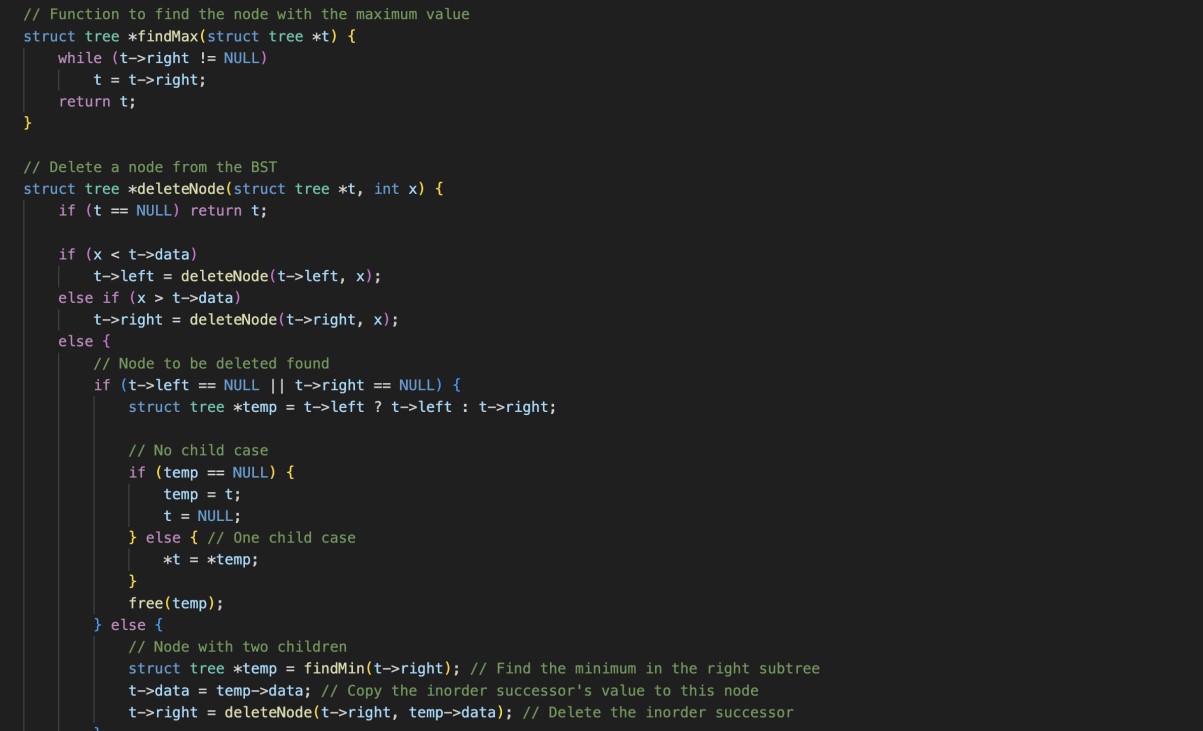
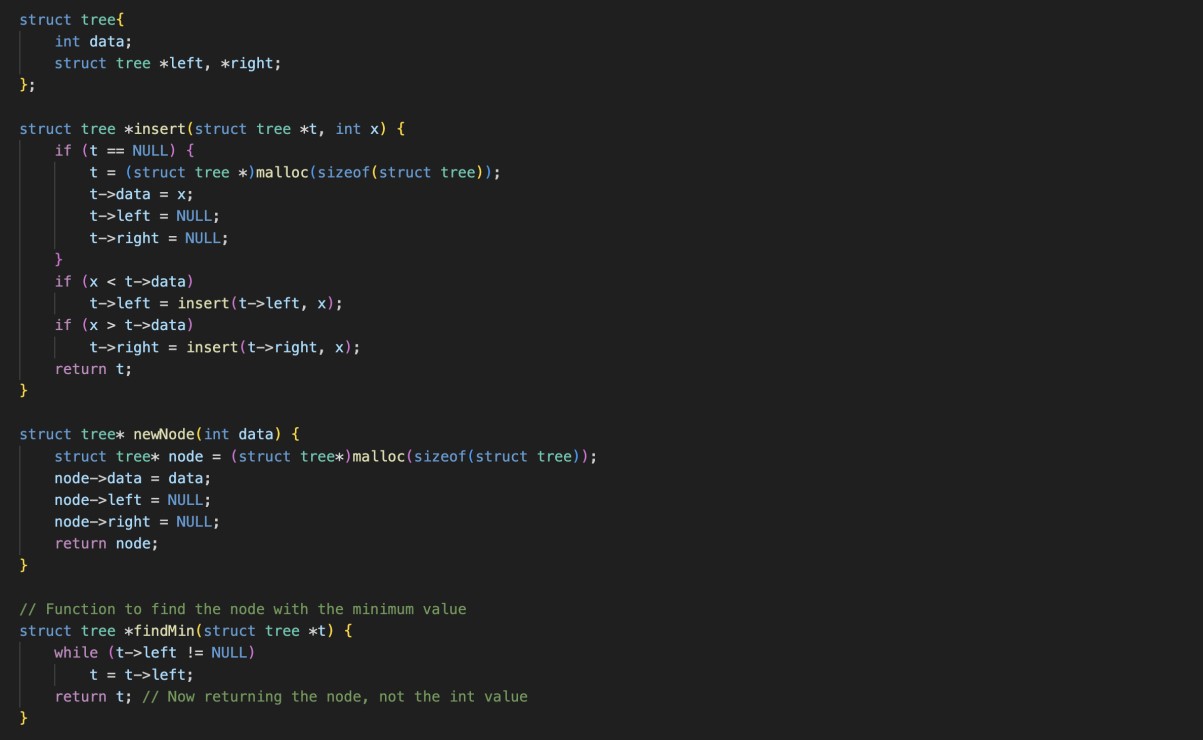


Write a function to check whether the given tree is BST

**Best practices to be followed:**

* Design before coding
* Usage of algorithm notation
* Use of multi-file C program
* Versioning of code





Technical Outcomes

|  |  |  |
| --- | --- | --- |
| Design | 2 | Needs improvement |
| Understanding of DS | 2 | Needs improvement |
| Use of DS | 3 |  |
| Debugging | 3 |  |

Best Practices

|  |  |  |
| --- | --- | --- |
| Design before coding | 2 | Needs improvement |
| Usage of Algo | 3 |  |
| Multifile | 1 | Needs improvement |
| Versioning | 3 |  |