Birla Institute of Technology and Science, Pilani Hyd Campus
CS F211: Data Structures and Algorithms
2nd Semester 2024-25 Lab No: 12
BST, and AVL Trees (7th-11th April 2025)

Program 1: (Prog1.cpp attached) BST:

Given a Partially Complete Binary Search Tree (Prog1.cpp) code in this program. here are your tasks:

- Task 1: Implement insert() method to insert an element into the Binary Search Tree. Part code is given for you. Complete the while loop within the insert method.
- Task 2: Implement search() method to return the pointer to the node containing the given element or NULL.

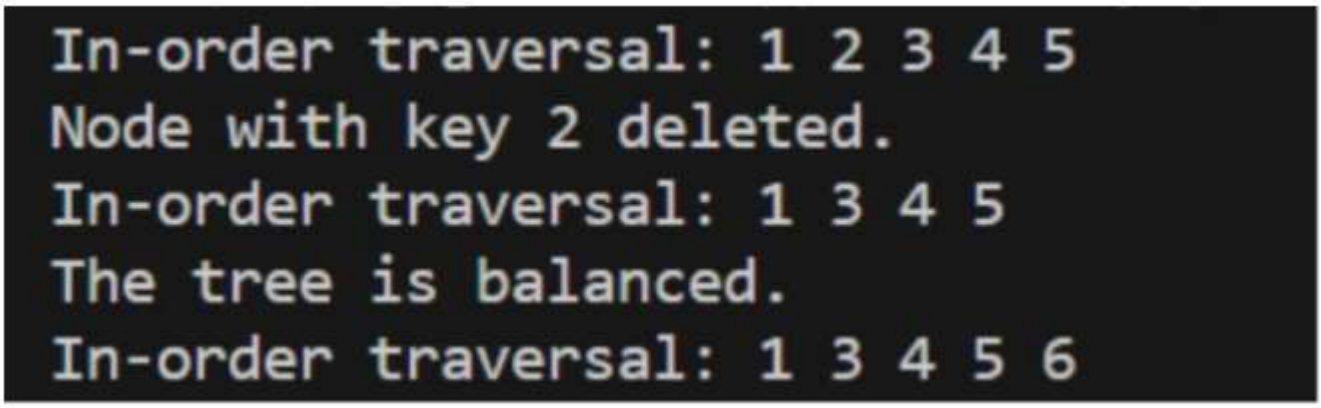
You may choose to go with either Iterative or Recursive code to solve these tasks. The driver code is already available. You should get the output as shown below:

Sample Run:

```
[1] Insert an element.
[2] Remove an element.
[3] Search an element.
[4] Display Inorder
[5] Height of BST
[6] Exit.
+------------------------------+
Enter your choice: 1
Enter an element to insert: 24

+------------------------------+
Enter your choice: 1
Enter an element to insert: 45

+------------------------------+
Enter your choice: 1
Enter an element to insert: 78

+------------------------------+
Enter your choice: 1
Enter an element to insert: 89

+------------------------------+
Enter your choice: 4
BST Elements:-
24 45 78 89

+------------------------------+
Enter your choice: 5
Height of Tree = 4

+------------------------------+
Enter your choice:
```

Check manually if it is a balanced BST? Run with few more test cases so that you can create a balanced (near) BST and do all these operations.

Program 2 (Prog2.cpp attached) AVL Tree: An AVL Tree is a self-balancing binary search tree (BST) that was designed to maintain a balanced structure, ensuring that the height of the tree remains logarithmic. You are given an implementation of AVL Tree in Prog2.cpp. Verify if you get the following output.



Task 3: Understanding Rotations: Modify the code to print messages whenever a left or right rotation occurs.

Task 4: Implement a function to find minimum and maximum keys in the AVL tree.

Task 5: Modify the search function to return the depth (level) at which a key is found.
