Experiment 6

Name:Sangam Raj UID:22BCS14875

Branch: CSE Section/Group: 640`B

Semester: 6 Date of Performance:10-03-25

Subject Name: AP LAB-II Subject Code: 22CSP-351

1. Aim:

a. To find and implement the maximum depth of Binary Tree.

b. To develop an algorithm for Binary Tree Inorder traversal.

2. Objective:

To implement and analyze maximum depth of Binary Tree. To develop an algorithm for Binary Tree Inorder traversal.

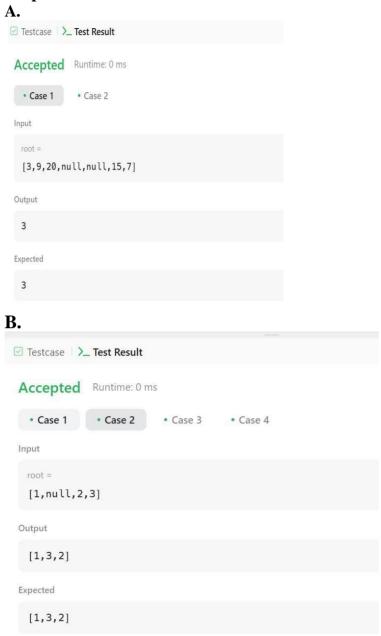
3. Implementation/Code:

```
A. class Solution {
  public int maxDepth(TreeNode root) {
     if (root == null) return 0; // Base case: If tree is empty
     int leftDepth = maxDepth(root.left); // Recursively find left subtree depth
     int rightDepth = maxDepth(root.right); // Recursively find right subtree depth
     return Math.max(leftDepth, rightDepth) + 1; // Return the maximum depth
         }
</>Code
                                                                      三 □ () □
Java ∨ 🔒 Auto
  1 class Solution {
        public int maxDepth(TreeNode root) {
            if (root == null) return 0; // Base case: If tree is empty
  4
            int leftDepth = maxDepth(root.left); // Recursively find left subtree depth
  5
  6
            int rightDepth = maxDepth(root.right); // Recursively find right subtree depth
  8
            return Math.max(leftDepth, rightDepth) + 1; // Return the maximum depth
 10
 11
```

```
B. import java.util.*;
    class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
    List<Integer> result = new ArrayList<>();
    inorderHelper(root, result);
    return result;
    private void inorderHelper(TreeNode node, List<Integer> result) {
    if (node == null) return;
    inorderHelper(node.left, result); // Visit left subtree
    result.add(node.val);
                               // Visit root
    inorderHelper(node.right, result); // Visit right subtree
     }
   }
</>Code
Java V
         Auto
       import java.util.*;
   1
   2
      class Solution {
   3
           public List<Integer> inorderTraversal(TreeNode root) {
  4 V
   5
                List<Integer> result = new ArrayList<>();
   6
                inorderHelper(root, result);
   7
                return result;
   8
           }
   9
  10
           private void inorderHelper(TreeNode node, List<Integer> result) {
                if (node == null) return;
  11
  12
  13
                inorderHelper(node.left, result); // Visit left subtree
  14
                result.add(node.val);
                                                        // Visit root
                inorderHelper(node.right, result); // Visit right subtree
  15
  16
           }
  17
  18
```

Saved

4. Output:



5. Learning Outcome:

- Understand string manipulation techniques in C++.
- Implement efficient algorithms for detecting cyclic rotations.
- Apply mathematical approaches to solve missing number problems.
- Utilize standard library functions like accumulate and find.
- Enhance problem-solving skills through algorithm design and analysis.