



Experiment 6

Student Name: Abhishek Roushan

UID: 22BCS10187

Branch: BE-CSE

Section/Group: 22BCS_NTPP_IOT_603

Semester: 6th

Date of Performance: 27-02-25

Subject Name: AP Lab-2

Subject Code: 22CSP-351

1. Aim: Symmetric Tree

2. Objective:

Given the root of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

3. Implementation/Code:

```
bool help(TreeNode* r1 , TreeNode* r2){  
  
    if(r1 == NULL and r2 == NULL){  
        return true;  
    }  
    if(r1 == NULL and r2 != NULL){  
        return false;  
    }  
    if(r1 != NULL and r2 == NULL){  
        return false;  
    }  
    if(r1->val != r2->val){  
        return false;  
    }  
  
    bool one = help(r1->left,r2->right);  
    bool two = help(r1->right,r2->left);  
  
    bool ans = one & two;  
  
    return ans;  
}  
bool isSymmetric(TreeNode* root) {  
  
    return help(root->left,root->right);  
  
}
```



4. Output

Accepted Runtime: 0 ms

- Case 1
- Case 2

Input

```
root =  
[1,2,2,3,4,4,3]
```

Output

```
true
```

Expected

```
true
```

Accepted Runtime: 0 ms

- Case 1
- Case 2

Input

```
root =  
[1,2,2,null,3,null,3]
```

Output

```
false
```

Expected

```
false
```

5. Learning Outcome:

- i. We Learn About the use of Recursion.
- ii. We Learn About the use of ListNode.
- iii. We Learn About the use of Base Cases.
- iv. We learn About the Calling function in recc..

Question 2

1. Aim:- Kth Smallest Element in a BST

2. Objective:-

Given the root of a binary search tree, and an integer k, return *the kth smallest value (**1-indexed**) of all the values of the nodes in the tree.*

3. Implementation/Code:-

```
void help(TreeNode* root,vector<int>&ans){

    if(root == NULL){
        return;
    }
    help(root->left,ans);
    ans.push_back(root->val);
    help(root->right,ans);
}

int kthSmallest(TreeNode* root, int k) {
    vector<int>ans;
    help(root,ans);

    return ans[k-1];
}
```

4. Output:-

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

root =
[3,1,4,null,2]

k =
1

Output

1

Expected

1

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

root =
[5,3,6,2,4,null,null,1]

k =
3

Output

3

Expected

3

5. Learning Outcome:

1. We Learn about the inorder traversal
2. We Learn about the function calls
3. We learned about recursion.

Question 3

6. Aim:- Convert Sorted Array to Binary Search Tree

7. Objective:-

Given an integer array `nums` where the elements are sorted in **ascending order**, convert *it to a **height-balanced** binary search tree*.

8. Implementation/Code:-

```
TreeNode*help(int s , int e , vector<int>&nums){

    if(s> e){
        return NULL;
    }

    int mid = (s+e)/2;
    TreeNode*newnode = new TreeNode(nums[mid]);

    newnode->left = help(s,mid-1,nums);
    newnode->right = help(mid+1,e,nums);
    return newnode;
}
TreeNode* sortedArrayToBST(vector<int>& nums) {
    int s =0;
    int e = nums.size()-1;
    return help(s, e, nums);
}
```

9. Output:-

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

nums =
[-10,-3,0,5,9]

Output

[0,-10,5,null,-3,null,9]

Expected

[0,-3,9,-10,null,5]

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

nums =
[1,3]

Output

[1,null,3]

Expected

[3,1]



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

10. Learning Outcome:

- We learn about to create a new node.
- We learn about function calls.
- We learn about the to push middle value.
- We learn to make a tree from recc.