

## LAB-7

1) Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

**code:**

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
struct BTreeNode
```

```
{
```

```
    int data;
```

```
    struct BTreeNode *left;
```

```
    struct BTreeNode *right;
```

```
};
```

```
struct BTreeNode* newNodeCreate(int value)
```

```
{
```

```
    struct BTreeNode* temp= (struct BTreeNode*)malloc(sizeof(struct BTreeNode));
```

```
    temp->data = value;
```

```
    temp->left = temp->right = NULL;
```

```
    return temp;
```

```
}
```

```
struct BTreeNode* insertNode(struct BTreeNode* node, int value)
```

```
{
```

```
    if (node == NULL)
```

```

{
    return newNodeCreate(value);
}

if (value < node->data)
{
    node->left = insertNode(node->left, value);
}

else if (value > node->data)
{
    node->right = insertNode(node->right, value);
}

return node;
}

void postOrder(struct BTreeNode* root)
{
    if (root != NULL)
    {
        postOrder(root->left);
        postOrder(root->right);
        printf(" %d ", root->data);
    }
}

void inOrder(struct BTreeNode* root)
{

```

```

    if (root != NULL)
    {
        inOrder(root->left);
        printf(" %d ", root->data);
        inOrder(root->right);
    }
}

void preOrder(struct BTreeNode* root)
{
    if (root != NULL)
    {
        printf(" %d ", root->data);
        preOrder(root->left);
        preOrder(root->right);
    }
}

void main()
{
    struct BTreeNode *root=NULL;
    root = insertNode(root, 50);
    insertNode(root, 30);
    insertNode(root, 20);
    insertNode(root, 40);
    insertNode(root, 70);
    insertNode(root, 60);
}

```

```

insertNode(root, 80);

printf("POSTORDER:");

postOrder(root);

printf("\n");

printf("PREORDER:");

preOrder(root);

printf("\n");

printf("INORDER:");

inOrder(root);

printf("\n");
}

```

#### **OUTPUT:**

POSTORDER: 20 40 30 60 80 70 50

PREORDER: 50 30 20 40 70 60 80

INORDER: 20 30 40 50 60 70 80

#### 2)LEETCODE:

Invert Binary Tree:

CODE:

```

struct TreeNode* invertTree(struct TreeNode* root) {

    if(root == NULL){

        return root;

    }

    invertTree(root->left);

    invertTree(root->right);

```

```

struct TreeNode* temp = root->left;

root->left = root->right;

root->right = temp;

return root;
}

```

226. Invert Binary Tree

Given the `root` of a binary tree, invert the tree, and return its `root`.

**Example 1:**

Input: `root = [4,2,7,1,3,6,9]`  
Output: `[4,7,2,9,6,3,1]`

**Example 2:**

Input: `root = [2,1,3]`  
Output: `[2,3,1]`

**Example 3:**

Input: `root = []`  
Output: `[]`

```

1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     struct TreeNode *left;
6  *     struct TreeNode *right;
7  * };
8  */
9 struct TreeNode* invertTree(struct TreeNode* root) {
10     if(root == NULL){
11         return root;
12     }
13     invertTree(root->left);
14     invertTree(root->right);
15     struct TreeNode* temp = root->left;
16     root->left = root->right;
17     root->right = temp;
18     return root;
19 }

```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: `root = [4,2,7,1,3,6,9]`

Output: `[4,7,2,9,6,3,1]`

Expected: `[4,7,2,9,6,3,1]`

Accepted Shivaraj\_K\_Pujari submitted on Feb 19, 2024 12:50

Runtime: 0 ms Memory: 6.00 MB Beats 100.00% of users with C Beats 17.20% of users with C

```

1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     struct TreeNode *left;
6  *     struct TreeNode *right;
7  * };
8  */
9 struct TreeNode* invertTree(struct TreeNode* root) {
10     if(root == NULL){
11         return root;
12     }
13     invertTree(root->left);
14     invertTree(root->right);
15     struct TreeNode* temp = root->left;
16     root->left = root->right;
17     root->right = temp;
18     return root;
19 }

```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: `root = [4,2,7,1,3,6,9]`

Output: `[4,7,2,9,6,3,1]`

Expected: `[4,7,2,9,6,3,1]`

Shivaraj\_K\_Pujari Access all features with our Premium subscription!

My Lists Notebook Submissions Progress Points Settings

Try New Features Orders My Playgrounds Revert to old version Appearance Sign Out

More challenges

2415. Reverse Odd Levels of Binary Tree

Write your notes here

