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 BTnode
{
   int data;
   struct BTnode *left;
   struct BTnode *right;
};

struct BTnode* newNodeCreate(int value)
{
   struct BTnode* temp= (struct BTnode*)malloc(sizeof(struct BTnode));
   temp->data = value;
   temp->left = temp->right = NULL;
   return temp;
}
struct BTnode* insertNode(struct BTnode* 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 BTnode* root)
{

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

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

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