DSA lab

**Assignment 7**

**Name:** Madhav Jha

**Roll no.:** E3-48

**Branch:** CSE (AI & ML)

# Binary Search Tree

## Statement:

Implement a Binary search tree with operations – create, search, insert, inorder, preorder and postorder. Write a menu driven program that performs the above operations also counts the total nodes and total leaf nodes in the tree. int count(T) – returns the total number of nodes from BST int countLeaf(T) – returns the total number of leaf nodes from BST

## Code:

#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

    int val;

    struct TreeNode\* left;

    struct TreeNode\* right;

};

void addNode(struct TreeNode\*\* root, int num) {

    struct TreeNode\* node;

    node = (struct TreeNode\*)malloc(sizeof(struct TreeNode));

    node->val = num;

    node->left = NULL;

    node->right = NULL;

    if (\*root == NULL) {

        \*root = node;

        return;

    }

    struct TreeNode\* temp = \*root;

    while (temp != NULL) {

        if (temp->val == num) {

            break;

        }if (temp->val < num) {

            if (temp->right != NULL) {

                temp = temp->right;

            }

            else {

                temp->right = node;

                break;

            }

        }

        else {

            if (temp->left != NULL) {

                temp = temp->left;

            }

            else {

                temp->left = node;

                break;

            }

        }

    }

}

void preOrder(struct TreeNode\* node) {

    if (node == NULL) return;

    printf("%d ", node->val);

    preOrder(node->left);

    preOrder(node->right);

}

void inOrder(struct TreeNode\* node) {

    if (node == NULL) return;

    inOrder(node->left);

    printf("%d ", node->val);

    inOrder(node->right);

}

void postOrder(struct TreeNode\* node) {

    if (node == NULL) return;

    postOrder(node->left);

    postOrder(node->right);

    printf("%d ", node->val);

}

int searchBST(struct TreeNode\*\* root, int num) {

    struct TreeNode\* temp = \*root;

    while (temp != NULL) {

        if (temp->val == num) {

            return 1;

        }

        else if (temp->val < num) {

            temp = temp->right;

        }

        else {

            temp = temp->left;

        }

    }

    return 0;

}

void countNodes(struct TreeNode\* node, int\* num) {

    if (node == NULL) return;

    \*num = \*num + 1;

    countNodes(node->left, num);

    countNodes(node->right, num);

}

int countAllNodes(struct TreeNode\*\* root) {

    int count = 0;

    countNodes(\*root, &count);

    return count;

}

void countLeaf(struct TreeNode\* node, int\* num) {

    if (node == NULL) return;

    if (node->left == NULL && node->right == NULL) {

        \*num = \*num + 1;

        return;

    }

    countLeaf(node->left, num);

    countLeaf(node->right, num);

}

int countAllLeaf(struct TreeNode\*\* root) {

    int count = 0;

    countLeaf(\*root, &count);

    return count;

}

int main(void) {

    struct TreeNode\*\* root = (struct TreeNode\*\*)malloc(sizeof(struct TreeNode\*));

    \*root = NULL;

    int choice = 0;

    do {

        printf("\n0. Enter 0 to exit!");

        printf("\n1. Add node to tree.");

        printf("\n2. Show Pre-order traversal.");

        printf("\n3. Show In-order traversal.");

        printf("\n4. Show Post-order traversal.");

        printf("\n5. Count all nodes.");

        printf("\n6. Count leafs");

        printf("\n7. Search number in BST");

        printf("\nYour choice: ");

        scanf("%d", &choice);

        int inp, src;

        switch (choice) {

        case 0:

            printf("\n\nExit...!\n\n");

            break;

        case 1:

            printf("\n\nEnter number to add: ");

            scanf("%d", &inp);

            addNode(root, inp);

            printf("\nNode added!!\n");

            break;

        case 2:

            printf("\n\nPre-order: ");

            preOrder(\*root);

            printf("\n\n");

            break;

        case 3:

            printf("\n\nIN-order: ");

            inOrder(\*root);

            printf("\n\n");

            break;

        case 4:

            printf("\n\nPost-order: ");

            postOrder(\*root);

            printf("\n\n");

            break;

        case 5:

            printf("\n\nAll nodes count: %d\n", countAllNodes(root));

            break;

        case 6:

            printf("\n\nAll leaf nodes count: %d\n", countAllLeaf(root));

            break;

        case 7:

            printf("\n\nEnter number to search: ");

            scanf("%d", &inp);

            if (searchBST(root, inp) == 0) {

                printf("\nNumber doesn't exist!!");

            }

            else {

                printf("\nNumber does exists!!");

            }

            printf("\n\n");

            break;

        default:

            printf("\nERROR: Invalid choice!!!");

            break;

        }

    } while (choice != 0);

    return 0;

}

## Output:













