1BM22CS242 Data Structure lab 7– 19-02-2024

8) Write a program

a) To construct a binary Search tree.

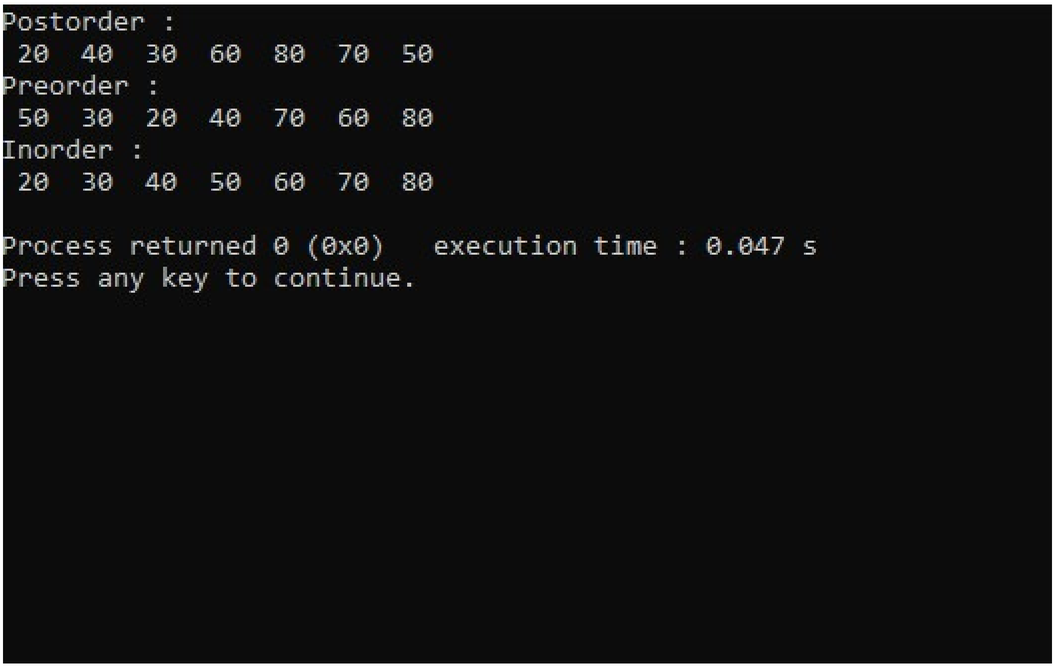
b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

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

struct Node{
    int data;
    struct Node *left, *right;
};
struct Node* newnode(int value)
{
    struct Node* temp= (struct Node*)malloc(sizeof(struct Node));
    temp->data = value;
    temp->left = temp->right = NULL;
    return temp;
}
struct Node* insertNode(struct Node* node, int value)
{
    if (node == NULL) {
        return newnode(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 Node* root)
{
    if (root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        printf(" %d ", root->data);
    }
}

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

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

int main()
{
    struct Node* 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 :\n");
    postOrder(root);
    printf("\n");
    printf("Preorder :\n");
    preOrder(root);
    printf("\n");

    printf("Inorder :\n");
    inOrder(root);
    printf("\n");

    return 0;
}
```

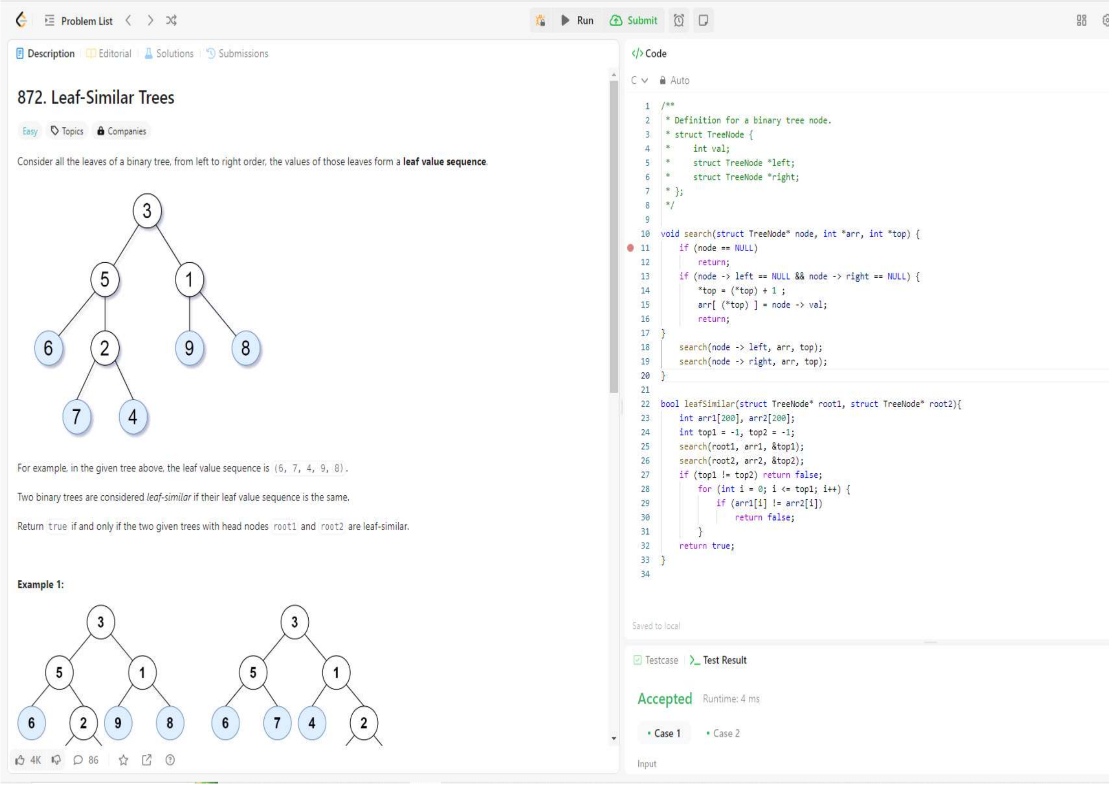
Output:



```
Postorder :
 20  40  30  60  80  70  50
Preorder :
 50  30  20  40  70  60  80
Inorder :
 20  30  40  50  60  70  80

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

Leetcode : Binary Search Tree

```
void search(struct TreeNode* node, int *arr, int *top) {
    if (node == NULL)
        return;
    if (node -> left == NULL && node -> right == NULL) {
        *top = (*top) + 1 ;
        arr[ (*top) ] = node -> val;
        return;
}
    search(node -> left, arr, top);
    search(node -> right, arr, top);
}

bool leafSimilar(struct TreeNode* root1, struct TreeNode* root2){
    int arr1[200], arr2[200];
    int top1 = -1, top2 = -1;
    search(root1, arr1, &top1);
    search(root2, arr2, &top2);
    if (top1 != top2) return false;
        for (int i = 0; i <= top1; i++) {
            if (arr1[i] != arr2[i])
                return false;
        }
    return true;
}
```



872. Leaf-Similar Trees

Consider all the leaves of a binary tree, from left to right order, the values of those leaves form a **leaf value sequence**.

For example, in the given tree above, the leaf value sequence is `(6, 7, 4, 9, 8)`.

Two binary trees are considered *leaf-similar* if their leaf value sequence is the same.

Return `true` if and only if the two given trees with head nodes `root1` and `root2` are leaf-similar.

Example 1:

```c
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */

void search(struct TreeNode* node, int *arr, int *top) {
    if (node == NULL)
        return;
    if (node -> left == NULL && node -> right == NULL) {
        *top = (*top) + 1 ;
        arr[ (*top) ] = node -> val;
        return;
}
    search(node -> left, arr, top);
    search(node -> right, arr, top);
}

bool leafSimilar(struct TreeNode* root1, struct TreeNode* root2){
    int arr1[200], arr2[200];
    int top1 = -1, top2 = -1;
    search(root1, arr1, &top1);
    search(root2, arr2, &top2);
    if (top1 != top2) return false;
        for (int i = 0; i <= top1; i++) {
            if (arr1[i] != arr2[i])
                return false;
        }
    return true;
}
```

Accepted Runtime: 4 ms