Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING

AND INFORMATION SYSTEMS

FALL SEMESTER 2024-2025

PMCA501P – DATA STRUCTURES AND ALGORITHM LAB

LAB ASSESSMENT – 3

SUBMITTED ON: 15 – NOV – 2024

SUBMITTED BY-

AKASH KUMAR BANIK

PROGRAM: MCA

REGISTER No.: 24MCA0242

1. Perform the following operations in a Binary Search Tree

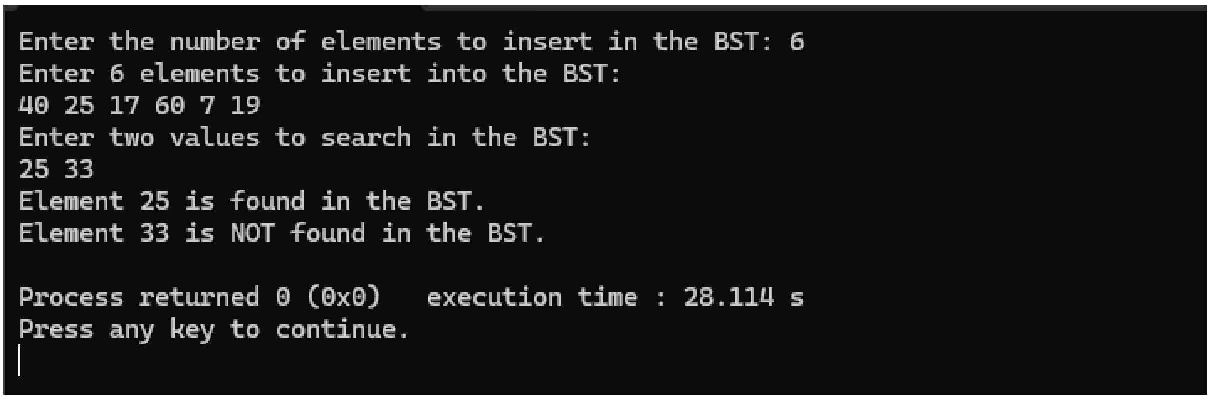
a) Create and insert the elements 40,25,17,60,7,19.

b) Search whether 25 and 33 are found or not (Text based output is enough).

CODE:

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

struct Node {
    int data;
    struct Node *left, *right;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct Node* insert(struct Node* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }
    return root;
}

int search(struct Node* root, int key) {
    if (root == NULL) {
        return 0;
    }
    if (root->data == key) {
        return 1;
    } else if (key < root->data) {
        return search(root->left, key);
    } else {
        return search(root->right, key);
    }
}

int main() {
    struct Node* root = NULL;
    int n, value, searchValue;

    printf("Enter the number of elements to insert in the BST: ");
    scanf("%d", &n);
    printf("Enter %d elements to insert into the BST:\n", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &value);

        root = insert(root, value);

    }

    printf("Enter two values to search in the BST:\n");

    for (int i = 0; i < 2; i++) {

        scanf("%d", &searchValue);

        if (search(root, searchValue)) {

            printf("Element %d is found in the BST.\n", searchValue);

        } else {

            printf("Element %d is NOT found in the BST.\n", searchValue);

        }

    }

    return 0;

}
```

OUTPUT:



```
Enter the number of elements to insert in the BST: 6
Enter 6 elements to insert into the BST:
40 25 17 60 7 19
Enter two values to search in the BST:
25 33
Element 25 is found in the BST.
Element 33 is NOT found in the BST.

Process returned 0 (0x0)   execution time : 28.114 s
Press any key to continue.
```

2. Assume that you are going to receive two binary trees as input. You need to merge these two binary trees into a new binary tree. The merge rule is that if two nodes overlap then sum the node values in the corresponding positions of the same level as the new resulting value of the merged tree. The merging process need to be start from the root nodes of both the input trees. Implement this using appropriate linked list or using arrays.

Input 1: 1 3 2 5

Input 2: 2 1 3 0 4 0 7

Here input 0 indicates the absence of either appropriate left or right child.

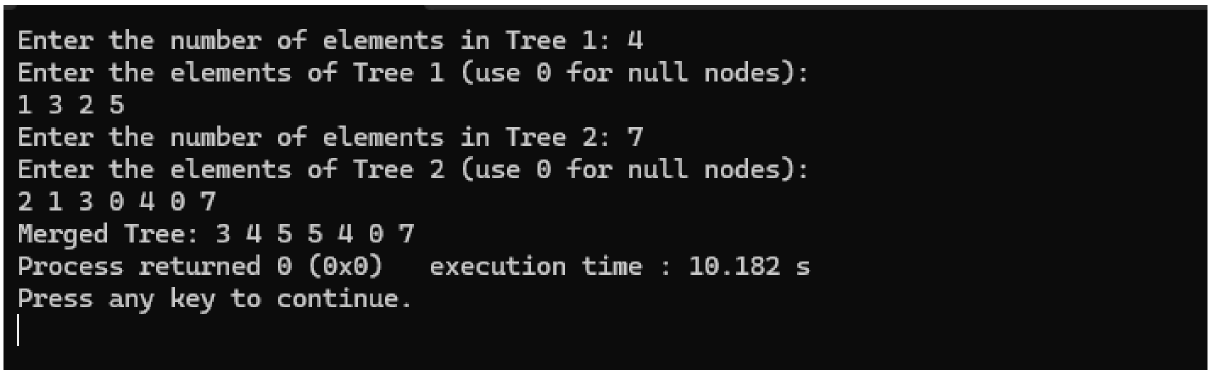
Output: 3 4 5 5 4 0 7 (Text based output is enough)

CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

void mergeTrees(int tree1[], int size1, int tree2[], int size2, int mergedTree[], int mergedSize)
{

    for (int i = 0; i < mergedSize; i++) {

        int val1 = (i < size1) ? tree1[i] : 0;

        int val2 = (i < size2) ? tree2[i] : 0;

        if (val1 == 0 && val2 == 0) {

            mergedTree[i] = 0;

        } else {

            mergedTree[i] = val1 + val2;

        }

    }

}
```

```
int main() {
    int size1, size2;
    printf("Enter the number of elements in Tree 1: ");
    scanf("%d", &size1);
    int tree1[size1];
    printf("Enter the elements of Tree 1 (use 0 for null nodes):\n");
    for (int i = 0; i < size1; i++) {
        scanf("%d", &tree1[i]);
    }

    printf("Enter the number of elements in Tree 2: ");
    scanf("%d", &size2);
    int tree2[size2];
    printf("Enter the elements of Tree 2 (use 0 for null nodes):\n");
    for (int i = 0; i < size2; i++) {
        scanf("%d", &tree2[i]);
    }
    int mergedSize = fmax(size1, size2);
    int mergedTree[mergedSize];
    for (int i = 0; i < mergedSize; i++) {
        mergedTree[i] = 0;
    }
    mergeTrees(tree1, size1, tree2, size2, mergedTree, mergedSize);
    printf("Merged Tree: ");
    for (int i = 0; i < mergedSize; i++) {
        printf("%d ", mergedTree[i]);
    }
    return 0;
}
```

OUTPUT:



```
Enter the number of elements in Tree 1: 4
Enter the elements of Tree 1 (use 0 for null nodes):
1 3 2 5
Enter the number of elements in Tree 2: 7
Enter the elements of Tree 2 (use 0 for null nodes):
2 1 3 0 4 0 7
Merged Tree: 3 4 5 5 4 0 7
Process returned 0 (0x0)   execution time : 10.182 s
Press any key to continue.
```