Rajalakshmi Engineering College

*241901104*

*241901104*

Name: Shafraz.R

*241901104*

*241901104*

Email: 241901104@rajalakshmi.edu.in

Roll no: 241901104

Phone: 6380360049

Branch: REC

Department: l CSE (CS) FB

Batch: 2028

Degree: B.E - CSE (CS)

NeoColab\_REC\_CS23231\_DATA STRUCTURES

*241901104*

*241901104*

REC\_DS using C\_Week 5\_COD\_Question 3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

*241901104*

*241901104*

*241901104*

*241901104*

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

*Input Format*

*241901104*

*241901104*

The first line of input consists of an integer n, representing the number of nodes in the binary search tree.

*241901104*

*241901104*

integers.

*241901104*

*241901104*

The second line consists of the values of the nodes, separated by space as

*241901104*

The third line consists of an integer representing, the value that is to be searched.

*241901104*

*Output Format*

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

Input: 7

*241901104*

*241901104*

*241901104*

*241901104*

*Sample Test Case*

8 3 10 1 6 14 23 6

Output: Value 6 is found in the tree.

*Answer*

// You are using GCC

*241901104*

*241901104*

*241901104*

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

*241901104*

struct Node {

int data; struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int value) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = value;

*241901104*

*241901104*

newNode->left = newNode->right = NULL; return newNode;

}

*241901104*

*241901104*

struct Node\* insert(struct Node\* root, int value) {

*241901104*

*241901104*

if (root == NULL) return createNode(value);

*241901104*

*241901104*

if (value < root->data) root->left = insert(root->left, value); else if (value > root->data) root->right = insert(root->right, value);

return root;

}

int search(struct Node\* root, int key) { if (root == NULL) return 0;

*241901104*

*241901104*

*241901104*

*241901104*

if (root->data == key) return 1; else if (key < root->data) return search(root->left, key); else

return search(root->right, key);

}

*241901104*

*241901104*

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

*241901104*

*241901104*

scanf("%d", &n); for (int i = 0; i < n; i++) { scanf("%d", &value);

root = insert(root, value);

}

scanf("%d", &searchValue);

else

*241901104*

*241901104*

if (search(root, searchValue))

printf("Value %d is found in the tree.\n", searchValue); printf("Value %d is not found in the tree.\n", searchValue);

*241901104*

*241901104*

return 0;

}

*241901104*

*241901104*

*241901104*

*241901104*

*Status :* Correct*Marks : 10/10*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*

*241901104*