Rajalakshmi Engineering College

Name: Harshavarthini
Email: 240701181@rajalakshmi.edu.in
Roll no: 240701181
Phone: 9150394958
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John, a computer science student, is learning about binary search trees (BST) and their properties. He decides to write a program to create a BST, display it in post-order traversal, and find the minimum value present in the tree.

Help him by implementing the program.

Input Format

The first line of input consists of an integer N, representing the number of elements to insert into the BST.

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

Output Format

The first line of output prints the space-separated elements of the BST in post-order traversal.

The second line prints the minimum value found in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 10 15
Output: 15 10 5
The minimum value in the BST is: 5

Answer

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

struct Node {
    int data;
    struct Node* left;
    struct Node* 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;
}
```

```
// You are using GCC
#include <iostream>
using namespace std;

// Node structure for the BST
struct Node {
    int data;
    Node* left;
```

```
    Node* right;

    Node(int val) {
        data = val;
        left = right = nullptr;
    }
};

// Function to insert a value into the BST
Node* insert(Node* root, int val) {
    if (root == nullptr)
        return new Node(val);

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

    return root;
}

// Function for post-order traversal: Left -> Right -> Root
void postOrder(Node* root) {
    if (root == nullptr)
        return;

    postOrder(root->left);
    postOrder(root->right);
    cout << root->data << " ";
}

// Function to find the minimum value in the BST (leftmost node)
int findMin(Node* root) {
    Node* current = root;
    while (current && current->left != nullptr) {
        current = current->left;
    }
    return current->data;
}

int main() {
    int N;
```

```
    cin >> N;

    Node* root = nullptr;

    for (int i = 0; i < N; ++i) {
        int val;
        cin >> val;
        root = insert(root, val);
    }

    // Output post-order traversal
    postOrder(root);
    cout << endl;

    // Output the minimum value in the BST
    int minValue = findMin(root);
    cout << "The minimum value in the BST is: " << minValue << endl;

    return 0;
}

int main() {
    struct Node* root = NULL;
    int n, data;
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        root = insert(root, data);
    }

    displayTreePostOrder(root);
    printf("\n");

    int minValue = findMinValue(root);
    printf("The minimum value in the BST is: %d", minValue);

    return 0;
}
```

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