Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
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 : 0

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;
}
```

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

// Define the structure for a tree node
struct Node {
    int value;
    struct Node* left;
    struct Node* right;
```

```
};

// Function to insert a value into the BST
struct Node* insert(struct Node* root, int data) {
    // If the tree is empty, create a new node
    if (root == NULL) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->value = data;
        newNode->left = NULL;
        newNode->right = NULL;
        return newNode;
    }

    // Otherwise, recur down the tree
    if (data < root->value) {
        root->left = insert(root->left, data); // Insert in the left subtree
    } else {
        root->right = insert(root->right, data); // Insert in the right subtree
    }

    return root; // Return the unchanged root pointer
}

// Function for post-order traversal of the BST
void displayTreePostOrder(struct Node* root) {
    if (root == NULL) {
        return; // Base case: if the node is NULL, do nothing
    }

    // Recursively traverse the left and right subtrees
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);

    // Print the current node's value
    printf("%d ", root->value);
}

// Function to find the minimum value in the BST
int findMinValue(struct Node* root) {
    if (root == NULL) {
        return -1; // Return -1 or some indication that the tree is empty
    }
```

```
    struct Node* current = root;

    // Traverse to the leftmost node
    while (current->left != NULL) {
        current = current->left;
    }

    return current->value; // Return the minimum value
}

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 : Wrong **Marks : 0/10**