Rajalakshmi Engineering College

Name: Jishnu Raj
Email: 241801109@rajalakshmi.edu.in
Roll no: 241801109
Phone: 9342455704
Branch: REC
Department: I AI & DS FB
Batch: 2028
Degree: B.E - AI & DS

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;
}


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

```
        root->right = insert(root->right, value);
    }
    return root;
}

// Display the BST in post-order traversal
void displayTreePostOrder(struct Node* root) {
    if (root == NULL) {
        return;
    }
    displayTreePostOrder(root->left);  // Traverse left subtree
    displayTreePostOrder(root->right); // Traverse right subtree
    printf("%d ", root->data);         // Print current node's value
}

// Find the minimum value in the BST
int findMinValue(struct Node* root) {
    if (root == NULL) {
        return -1; // Indicating the tree is empty, though the problem guarantees N >=
1.
    }
    while (root->left != NULL) {
        root = root->left; // The minimum value is in the leftmost node
    }
    return root->data;
}

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_**