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 2

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Mike is learning about Binary Search Trees (BSTs) and wants to implement various operations on them. He wants to write a basic program for creating a BST, inserting nodes, and printing the tree in the pre-order traversal.

Write a program to help him solve this program.

Input Format

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

The second line consists of N space-separated integers, representing the values to insert into the BST.

Output Format

The output prints the space-separated values of the BST in the pre-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 1 5 2 4

Output: 3 1 2 5 4

Answer

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

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;
    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 value) {
```

```
    // If the tree is empty, create a new node and return it
    if (root == NULL) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->value = value;
        newNode->left = NULL;
        newNode->right = NULL;
        return newNode;
    }

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

    return root; // Return the unchanged root pointer
}

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

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

    // Recursively print the left and right subtrees
    printPreorder(node->left);
    printPreorder(node->right);
}

// Main function to demonstrate the BST
int main() {
    struct Node* root = NULL; // Initialize the root of the BST
    int N;

    // Read the number of values to insert
    scanf("%d", &N);

    // Read N space-separated integers and insert them into the BST
```

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

    // Print the pre-order traversal of the BST
    printPreorder(root);
    printf("\n"); // Print a newline after the output

    return 0;
}

int main() {
    struct Node* root = NULL;

    int n;
    scanf("%d", &n);

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

    printPreorder(root);
    return 0;
}
```

Status : Wrong ***Marks : 0/10***

240701215