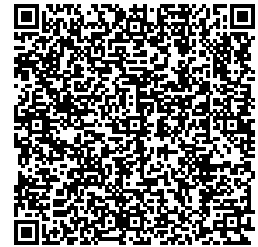


Rajalakshmi Engineering College

Name: VISHVA RAGAVAN P
Email: 241501249@rajalakshmi.edu.in
Roll no:
Phone: 6381117385
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

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 : 10

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;  
}
```

// Function to create a new node with the given value

```
struct Node* newNode(int value) {  
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));  
    node->data = value;  
    node->left = node->right = NULL;  
    return node;  
}
```

// 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  
    if (root == NULL) {
```

```

        return newNode(value);
    }

    // Otherwise, recur down the tree
    if (value < root->data) {
        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) node pointer
}

// Function to print the tree in pre-order traversal (root, left, right)
void printPreorder(struct Node* node) {
    if (node == NULL) {
        return;
    }
    // Print root, then left subtree, then right subtree
    printf("%d ", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}

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 : Correct

Marks : 10/10