Binary Search Tree

#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 = NULL;

    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 if (value > root->data)

    {

        root->right = insert(root->right, value);

    }

    return root;

}

void inorderTraversal(struct Node \*root)

{

    if (root != NULL)

    {

        inorderTraversal(root->left);

        printf("%d ", root->data);

        inorderTraversal(root->right);

    }

}

void preorderTraversal(struct Node \*root)

{

    if (root != NULL)

    {

        printf("%d ", root->data);

        preorderTraversal(root->left);

        preorderTraversal(root->right);

    }

}

void postorderTraversal(struct Node \*root)

{

    if (root != NULL)

    {

        postorderTraversal(root->left);

        postorderTraversal(root->right);

        printf("%d ", root->data);

    }

}

int main()

{

    struct Node \*root = NULL;

    int n, value;

    printf("Enter the number of nodes: ");

    scanf("%d", &n);

    printf("Enter %d values to insert into BST:\n", n);

    for (int i = 0; i < n; i++)

    {

        scanf("%d", &value);

        if (root == NULL)

        {

            root = insert(root, value);

        }

        else

        {

            insert(root, value);

        }

    }

    printf("\nInorder traversal of the BST: ");

    inorderTraversal(root);

    printf("\nPreorder traversal of the BST: ");

    preorderTraversal(root);

    printf("\nPostorder traversal of the BST: ");

    postorderTraversal(root);

    printf("\n");

    return 0;

}