#include <iostream>

using namespace std;

// Node structure for the binary tree

struct Node {

    int data;

    Node\* left;

    Node\* right;

};

// Function to create a new node

Node\* createNode(int value) {

    Node\* newNode = new Node();

    newNode->data = value;

    newNode->left = newNode->right = nullptr;

    return newNode;

}

// Function to insert a node into the binary tree

Node\* insertNode(Node\* root, int value) {

    if (root == nullptr)

        return createNode(value);

    if (value < root->data)

        root->left = insertNode(root->left, value);

    else if (value > root->data)

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

    return root;

}

// Function to perform inorder traversal recursively

void inorderTraversal(Node\* root) {

    if (root != nullptr) {

        inorderTraversal(root->left);

        cout << root->data << " ";

        inorderTraversal(root->right);

    }

}

// Function to perform preorder traversal recursively

void preorderTraversal(Node\* root) {

    if (root != nullptr) {

        cout << root->data << " ";

        preorderTraversal(root->left);

        preorderTraversal(root->right);

    }

}

// Function to perform postorder traversal recursively

void postorderTraversal(Node\* root) {

    if (root != nullptr) {

        postorderTraversal(root->left);

        postorderTraversal(root->right);

        cout << root->data << " ";

    }

}

int main() {

    Node\* root = nullptr;

    int n, value;

    cout << "Enter the number of nodes in the binary tree: ";

    cin >> n;

    cout << "Enter the values of the nodes: ";

    for (int i = 0; i < n; ++i) {

        cin >> value;

        root = insertNode(root, value);

    }

    cout << "Inorder traversal: ";

    inorderTraversal(root);

    cout << endl;

    cout << "Preorder traversal: ";

    preorderTraversal(root);

    cout << endl;

    cout << "Postorder traversal: ";

    postorderTraversal(root);

    cout << endl;

    return 0;

}