**#include <bits/stdc++.h>**

**using namespace std;**

**#define M 1000000007**

**struct node {**

**int data;**

**node \*left, \*right, \*parent;**

**node(int data) {**

**this->data = data;**

**this->left = nullptr;**

**this->right = nullptr;**

**this->parent = nullptr;**

**}**

**};**

**void insert(node\* root, int data) {**

**if(root->left == nullptr) {**

**node\* n = new node(data);**

**root->left = n;**

**return;**

**}**

**if(root->right == nullptr) {**

**node\* n = new node(data);**

**root->right = n;**

**return;**

**}**

**insert(root->left, data);**

**}**

**void preOrderRecursiveTraversal(node\* root) {**

**if(root != nullptr) {**

**cout<<root->data<<" ";**

**preOrderRecursiveTraversal(root->left);**

**preOrderRecursiveTraversal(root->right);**

**}**

**}**

**void postOrderRecursiveTraversal(node\* root) {**

**if(root!= nullptr) {**

**postOrderRecursiveTraversal(root->left);**

**postOrderRecursiveTraversal(root->right);**

**cout<<root->data<<" ";**

**}**

**}**

**void inOrderRecursiveTraversal(node\*root) {**

**if(root != nullptr) {**

**inOrderRecursiveTraversal(root->left);**

**cout<<root->data<<" ";**

**inOrderRecursiveTraversal(root->right);**

**}**

**}**

**void preorderIterativeTraversal(node\* root) {**

**stack<node\*>stk;**

**stk.push(root);**

**while(!stk.empty()) {**

**node\* element = stk.top();**

**stk.pop();**

**cout<<element->data<<" ";**

**if(element->right!=nullptr) {**

**stk.push(element->right);**

**}**

**if (element->left != nullptr) {**

**stk.push(element->left);**

**}**

**}**

**}**

**void inorderIterativeTraversal(node\* root) {**

**if(root == nullptr) {**

**return;**

**}**

**stack<node\*> stk;**

**while(root != nullptr or !stk.empty()) {**

**while(root != nullptr) {**

**stk.push(root);**

**root = root->left;**

**}**

**node\* n = stk.top();**

**stk.pop();**

**cout<<n->data<<" ";**

**if(n->right != nullptr) {**

**root = n->right;**

**} else {**

**root = nullptr;**

**}**

**}**

**}**

**int main() {**

**node\* root = new node(6);**

**insert(root, 7);**

**insert(root, 10);**

**insert(root, 12);**

**insert(root, 16);**

**preOrderRecursiveTraversal(root);**

**cout<<endl;**

**postOrderRecursiveTraversal(root);**

**cout<<endl;**

**inOrderRecursiveTraversal(root);**

**cout<<endl;**

**preorderIterativeTraversal(root);**

**cout<<endl;**

**inorderIterativeTraversal(root);**

**cout<<endl;**

**return 0;**

**}**