#include <iostream>

using namespace std;

class Node {

private:

int data;

Node\* right;

Node\* left;

public:

Node(int data) {

this->data = data;

this->right = NULL;

this->left = NULL;

}

void setData(int data) {

this->data = data;

}

void setRight(Node\* right) {

this->right = right;

}

void setLeft(Node\* left) {

this->left = left;

}

int getData() {

return data;

}

Node\* getRight() {

return right;

}

Node\* getLeft() {

return left;

}

};

class BST {

private:

Node\* root;

public:

BST()

{

this->root = NULL;

}

void insert(int data)

{

root = insert(root, data);

}

void InOrder\_display()

{

InOrder\_display(root);

cout << endl;

}

void PreOrder\_display()

{

PreOrder\_display(root);

cout << endl;

}

void PostOrder\_display()

{

PostOrder\_display(root);

cout << endl;

}

bool search(int data)

{

return searchRec(root, data);

}

Node\* insert(Node\* root, int data)

{

if (root == NULL)

{

return new Node(data);

}

if (data < root->getData())

{

root->setLeft(insert(root->getLeft(), data));

}

else

{

root->setRight(insert(root->getRight(), data));

}

return root;

}

//Pre Order Display

void PreOrder\_display(Node\* node)

{

if (node != NULL)

{

cout << node->getData() << " ";

PreOrder\_display(node->getLeft());

PreOrder\_display(node->getRight());

}

}

//Inorder Display

void InOrder\_display(Node\* node)

{

if (node != NULL)

{

InOrder\_display(node->getLeft());

cout << node->getData() << " ";

InOrder\_display(node->getRight());

}

}

// Post Order Display

void PostOrder\_display(Node\* node)

{

if (node != NULL)

{

PostOrder\_display(node->getLeft());

PostOrder\_display(node->getRight());

cout << node->getData() << " ";

}

}

bool searchRec(Node\* node, int data)

{

if (node == NULL)

{

return false;

}

if (data == node->getData()) {

return true;

}

if (data < node->getData())

{

return searchRec(node->getLeft(), data);

}

else

{

return searchRec(node->getRight(), data);

}

}

};

int main()

{

BST bst;

bst.insert(10);

bst.insert(1);

bst.insert(3);

bst.insert(13);

bst.insert(2);

bst.insert(91);

bst.PreOrder\_display() ;

cout<<endl;

bst.InOrder\_display();

cout<<endl;

bst.PostOrder\_display() ;

return 0;

}