#include <iostream>

using namespace std;

struct node {

int data;

node \*left;

node \*right;

node(int val) {

data = val;

left = NULL;

right = NULL;

}

};

class BST {

node \*root;

public:

BST() {

root = NULL;

}

node \*insert(node \*root, int key) {

if (root == NULL) {

root = new node(key);

return root;

}

else if (key > root->data) {

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

}

else if (key < root->data) {

root->left = insert(root->left, key);

}

return root;

}

node \*create() {

cout << "Enter the total number of nodes: ";

int n;

cin >> n;

for (int i = 0; i < n; i++) {

cout << "Enter the data: ";

int val;

cin >> val;

root = insert(root, val);

}

return root;

}

int minelem(node \*root){

node \*temp = root;

while (temp->left != NULL){

temp = temp->left;

}

return temp->data;

}

int maxelem(node \*root){

node \*temp = root;

while (temp->right != NULL){

temp = temp->right;

}

return temp->data;

}

void in(node \*root){ //Used to print BST in sorted order(inorder traversal)

if (root == NULL) return;

in(root->left);

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

in(root->right);

}

int height(node \*root){

if (root == NULL) return 0;

return 1 + max(height(root->left), height(root->right)); //Adding 1 accounts for the current node itself

}

void elems(node \*root){

while (root != NULL){

if (height(root->left) > height(root->right)){

if (root->left != NULL)

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

root = root->left;

}else{

if (root->right != NULL)

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

root = root->right;

}

}

}

node \*search(int key, node \*root){

while (root != NULL) {

if (key == root->data){

return root;

} else if (key > root->data){

root = root->right;

} else{

root = root->left;

}

}

return NULL;

}

node \*swap(node \*root){

if (root == NULL) return NULL;

node \*temp = root->left;

root->left = swap(root->right);

root->right = swap(temp);

return root;

}

void menu(){

char ch;

do{

cout << "1.Create a BST\n2.Insert a Node\n3.Display BST\n4.Minimum Element\n5.Maximum Element\n6.Height of the Tree\n7.Swap the Nodes\n8.Search a Value\n9.Exit" << endl;

cout << "Enter your choice: ";

cin >> ch;

if (ch == '1') {

create();

}

else if (ch == '2') {

cout << "Enter the data of the node: ";

int data;

cin >> data;

root = insert(root, data);

}

else if (ch == '3') {

in(root);

cout << endl;

}

else if (ch == '4') {

int num = minelem(root);

cout << "The minimum element in the BST is: " << num << endl;

}

else if (ch == '5') {

int num = maxelem(root);

cout << "The maximum element in the BST is: " << num << endl;

}

else if (ch == '6') {

int h = height(root);

cout << "The height of the tree (number of nodes in longest path): " << h << endl;

cout << "Elements along longest path: ";

cout << endl;

elems(root);

}

else if (ch == '7') {

cout << "Tree after swapping: ";

root = swap(root);

in(root);

cout << endl;

}

else if (ch == '8') {

cout << "Enter value to search: ";

int val;

cin >> val;

node \*found = search(val, root);

if (found != NULL)

cout << "Value found in BST." << endl;

else

cout << "Value not found." << endl;

}

else if (ch == '9') {

break;

}

else {

cout << "Invalid Input" << endl;

}

} while (ch != '9');

}

};

int main() {

BST b;

b.menu();

return 0;

}