#include <iostream> using namespace std; class node

{

public:

int data; node \*left; node \*right;

};

class bst

{

public:

node \*root; bst()

{

root = NULL;

}

void create(); void insert();

void postorder(node \*); void inorder(node \*); void preorder(node \*); void search(int key);

int search (node\*, int key);

void minimum();

int height(node \*);

};

void bst::minimum()

{

node \*temp; int min; temp = root;

while (temp->left != NULL)

{

min = temp->data; temp = temp->left;

if (temp->data < min)

{

min = temp->data;

}

else

{

temp = temp->left;

}

}

cout << "minimum no. is:" << min;

}

int bst::height(node \*root)

{

if (root == NULL)

{

return 0;

}

else

{

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

{

return (1 + height(root->right));

}

else

{

return (1 + height(root->left));

}

}

}

void bst::create()

{

node \*curr, \*temp; int ans = 1;

cout << "enter data:"; do

{

curr = new node; cin >> curr->data;

curr->left = curr->right = NULL; if (root == NULL)

{

root = curr;

}

else

{

temp = root; while (1)

{

if (curr->data <= temp->data)

{

if (temp->left == NULL)

{

temp->left = curr; break;

}

else

{

temp = temp->left;

}

}

else

{

if (temp->right == NULL)

{

temp->right = curr; break;

}

else

{

temp = temp->right;

}

}

}

}

cout << "want to continue:"; cin >> ans;

} while (ans == 1);

}

void bst::inorder(node \*root)

{

if (root != NULL)

{

inorder(root->left);

cout << " " << root->data; inorder(root->right);

}

}

void bst::preorder(node \*root)

{

if (root != NULL)

{

cout << " " << root->data; preorder(root->left); preorder(root->right);

}

}

void bst::postorder(node \*root)

{

if (root != NULL)

{

postorder(root->left); postorder(root->right); cout << " " << root->data;

}

}

void bst::insert()

{

node \*curr, \*temp; int ans = 1;

cout << "enter data:";

curr = new node; cin >> curr->data;

curr->left = curr->right = NULL; if (root == NULL)

{

root = curr;

}

else

{

temp = root; while (1)

{

if (curr->data <= temp->data)

{

if (temp->left == NULL)

{

temp->left = curr; break;

}

else

{

temp = temp->left;

}

}

else

{

if (temp->right == NULL)

{

temp->right = curr; break;

}

else

{

temp = temp->right;

}

}

}

}

}

void bst::search(int key)

{

node \*curr; curr = root;

while (curr != NULL)

{

if (curr->data == key)

{

cout << "found"; break;

}

else

{

if (key < curr->data)

{

curr = curr->left;

}

else

{

curr = curr->right;

}

}

}

if (curr == NULL)

{

cout << "not found";

}

}

int main()

{

bst b;

int key, ch; do

{

cout << "\n1.create\n2.insert\n3.inorder\n4.preorder\n5.postorder\n6.search\n7.minimum\n8.height\np ress 0 to exit\n";

cout << "enter your choice:"; cin >> ch;

switch (ch)

{

case 1:

b.create(); break;

case 2:

b.insert(); break;

case 3:

cout << "inorder traversal is\n"; b.inorder(b.root);

break; case 4:

cout << "preorder traversal is\n"; b.preorder(b.root);

break; case 5:

cout << "postorder traversal is\n"; b.postorder(b.root);

break; case 6:

cout << "\nenter key:"; cin >> key; b.search(key);

break;

case 7:

b.minimum(); break;

case 8:

cout << "height of tree: " << b.height(b.root); break;

}

} while (ch != 0); return 0;

}