Given a list of integers, you will need to write a C++ program to construct a binary search tree (left sub tree is less than the right subtree). After constructing the binary search tree do an inorder traversal and preorder traversal. Find the positions where node value is same in both the positions.

This checking can be done by storing inorder traversal & preorder traversals of the tree in two separate arrays.

#include <iostream>

using namespace std;

struct Node{

int data;

Node \*left;

Node \*right;

Node(){ }

Node(int data){

this->data = data;

left = right = NULL;

}

};

class Tree{

Node \*root;

int \*inordert;

int \*preordert;

int count;

public:

Tree(){

root = NULL;

}

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

if( root == NULL)

return new Node(data);

else if( data > root->data)

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

else

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

return root;

}

void inorder(){

count = 0;

inorder(root);

cout<<endl;

}

void insert(int data){

root = insert(root, data);

}

void inorder(Node \*root){

if( root != NULL){

inorder( root->left);

inordert[count++] = root->data;

inorder( root->right);

}

}

void preorder(Node \*root){

if( root != NULL){

preordert[count++] = root->data;

preorder(root->left);

preorder(root->right);

}

}

void printordermatch(int n){

preordert = new int[n];

inordert = new int[n];

count = 0;

preorder(root);

count = 0;

inorder(root);

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

if( preordert[i] == inordert[i])

cout<<(i+1)<<" "<<preordert[i]<<endl;

}

};

int main(){

int n;

int count = 0;

Tree tree;

while( true ){

cin>>n;

// cout<<n<<endl;

if( n == -1)

break;

count++;

tree.insert(n);

}

tree.printordermatch(count);

return 0;

}