#include <iostream>

#include <time.h>

using namespace std;

class Tree {

private:

struct Node {

int data;

Node \*left, \*right;

};

Node \*root;

void show\_tree\_Pr(Node \*node) {

if (node != NULL) {

cout << node->data << " ";

show\_tree\_Pr(node->left);

show\_tree\_Pr(node->right);

}

}

void show\_tree\_Obr(Node \*node) {

if (node != NULL) {

show\_tree\_Obr(node->left);

show\_tree\_Obr(node->right);

cout << node->data << " ";

}

}

void show\_tree\_Sim(Node \*node) {

if (node != NULL) {

show\_tree\_Sim(node->left);

cout << node->data << " ";

show\_tree\_Sim(node->right);

}

}

public:

Tree() {

root = NULL;

}

void add(Node \*&root, int value) {

if (root == NULL) {

root = new Node;

root->data = value;

root->left = NULL;

root->right = NULL;

}

else {

if (rand() % 2) {

//cout << " l";

add(root->left, value);

}

else {

//cout << " r";

add(root->right, value);

}

}

}

void add(int value) {

add(root, value);

}

void show\_Pr() {

show\_tree\_Pr(root);

cout << endl;

}

void show\_Obr() {

show\_tree\_Obr(root);

cout << endl;

}

void show\_Sim() {

show\_tree\_Sim(root);

cout << endl;

}

bool DeleteElement(Node\* node) {

if (node != NULL) {

if (node->left != NULL) {

node->data = node->left->data;

if (DeleteElement(node->left)) {

node->left = NULL;

}

}

else if (node->right != NULL) {

node->data = node->right->data;

if (DeleteElement(node->right)) {

node->right = NULL;

}

}

else {

delete node;

return true;

}

}

return false;

}

bool DeleteEven(Node\* node) {

if (node != NULL) {

if (DeleteEven(node->left)) {

node->left = NULL;

}

if (DeleteEven(node->right)) {

node->right = NULL;

}

if (node->data % 2 == 0) {

if (DeleteElement(node)) {

return true;

}

}

}

return false;

}

void DeleteEven() {

DeleteEven(root);

}

};

void main() {

srand(time(0));

int TreeSize = rand() % 5 + 10;

Tree tree;

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

int v = rand() % 1000;

cout << v;

tree.add(v);

cout << endl;

}

cout << "Pre-order" << endl;

tree.show\_Pr();

cout << "Post-order" << endl;

tree.show\_Obr();

cout << "In-order" << endl;

tree.show\_Sim();

tree.DeleteEven();

cout << "Pre-order" << endl;

tree.show\_Pr();

system("pause");

}