#include <iostream>

using namespace std;

// Khai báo cấu trúc Node

struct Node {

    int data;

    Node\* left;

    Node\* right;

};

// Tạo node gốc cho cây

Node\* createNode(int data) {

    Node\* node = new Node();

    node->data = data;

    node->left = NULL;

    node->right = NULL;

    return node;

}

// Thêm vào node lá bên trái node p

void addLeft(Node\* p, int data) {

    if (p == NULL) {

        cout << "Khong ton tai node p!" << endl;

        return;

    }

    if (p->left != NULL) {

        cout << "Node p da co node con ben trai!" << endl;

        return;

    }

    p->left = createNode(data);

}

// Thêm vào node lá bên phải node p

void addRight(Node\* p, int data) {

    if (p == NULL) {

        cout << "Khong ton tai node p!" << endl;

        return;

    }

    if (p->right != NULL) {

        cout << "Node p da co node con ben phai!" << endl;

        return;

    }

    p->right = createNode(data);

}

// Loại bỏ node lá bên trái node p

void removeLeft(Node\* p) {

    if (p == NULL) {

        cout << "Khong ton tai node p!" << endl;

        return;

    }

    if (p->left == NULL) {

        cout << "Node p khong co node con ben trai!" << endl;

        return;

    }

    delete p->left;

    p->left = NULL;

}

// Loại bỏ node lá bên phải node p

void removeRight(Node\* p) {

    if (p == NULL) {

        cout << "Khong ton tai node p!" << endl;

        return;

    }

    if (p->right == NULL) {

        cout << "Node p khong co node con ben phai!" << endl;

        return;

    }

    delete p->right;

    p->right = NULL;

}

// Loại bỏ cả cây

void removeTree(Node\* root) {

    if (root == NULL) return;

    removeTree(root->left);

    removeTree(root->right);

    delete root;

}

// Tìm kiếm node trên cây

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

    if (root == NULL || root->data == data) return root;

    if (data < root->data) return search(root->left, data);

    return search(root->right, data);

}

// Duyệt cây theo thứ tự trước (Pre-order traversal)

void preOrder(Node\* root) {

    if (root == NULL) return;

    cout << root->data << " ";

    preOrder(root->left);

    preOrder(root->right);

}

// Duyệt cây theo thứ tự giữa (In-order traversal)

void inOrder(Node\* root) {

    if (root == NULL) return;

    inOrder(root->left);

    cout << root->data << " ";

    inOrder(root->right);

}

// Duyệt cây theo thứ tự sau (Post-order traversal)

void postOrder(Node\* root) {

    if (root == NULL) return;

    postOrder(root->left);

    postOrder(root->right);

    cout << root->data << " ";

}

int main() {

    Node\* root = createNode(1);

    addLeft(root, 2);

    addRight(root, 3);

    addLeft(root->left, 4);

    addRight(root->left, 5);

    addLeft(root->right, 6);

    addRight(root->right, 7);

    cout << "Duyet cay theo thu tu truoc: ";

    preOrder(root);

    cout << endl;

    cout << "Duyet cay theo thu tu giua: ";

    inOrder(root);

    cout << endl;

    cout << "Duyet cay theo thu tu sau: ";

    postOrder(root);

    cout << endl;

    Node\* p = search(root, 4);

    if (p == NULL) {

        cout << "Khong tim thay node can tim!" << endl;

    } else {

        cout << "Tim thay node co gia tri " << p->data << endl;

    }

    removeTree(root);

    system("pause");

return 0;

}

2. Bài tập ứng dụng

DSA11011 – Cây nhị phân hoàn hảo

#include <iostream>

#include <cmath>

using namespace std;

// Định nghĩa lớp Node để biểu diễn các nút trong cây

class Node {

public:

    int data;

    Node\* left;

    Node\* right;

};

// Tính chiều cao của cây

int height(Node\* root) {

    if (root == nullptr) {

        return 0;

    }

    int left\_height = height(root->left);

    int right\_height = height(root->right);

    return max(left\_height, right\_height) + 1;

}

// Kiểm tra xem cây nhị phân có phải là một cây hoàn hảo hay không

bool is\_perfect\_tree(Node\* root) {

    if (root == nullptr) {

        return true;

    }

    int left\_height = height(root->left);

    int right\_height = height(root->right);

    if (left\_height != right\_height) {

        return false;

    }

    return is\_perfect\_tree(root->left) && is\_perfect\_tree(root->right);

}

int main() {

    int t;

    cin >> t;

    while (t--) {

        int n;

        cin >> n;

        Node\* root = nullptr;

        Node\* nodes[n+1];

        for (int i = 1; i <= n; i++) {

            int u, v;

            char x;

            cin >> u >> v >> x;

            if (nodes[u] == nullptr) {

                nodes[u] = new Node();

                nodes[u]->data = u;

                nodes[u]->left = nullptr;

                nodes[u]->right = nullptr;

                if (root == nullptr) {

                    root = nodes[u];

                }

            }

            if (nodes[v] == nullptr) {

                nodes[v] = new Node();

                nodes[v]->data = v;

                nodes[v]->left = nullptr;

                nodes[v]->right = nullptr;

            }

            if (x == 'L') {

                nodes[u]->left = nodes[v];

            } else {

                nodes[u]->right = nodes[v];

            }

        }

        if (is\_perfect\_tree(root)) {

            cout << "Yes\n";

        } else {

            cout << "No\n";

        }

    }

    return 0;

}