**姓名: 张景耀 学号: 3200602123 班级: 计算机2005**

### 实验内容1: 对于一个不带头结点的单链表，设计递归算法逆置所有结点

#### 源代码

//1. 对于一个不带头结点的单链表，设计递归算法逆置所有结点

#include<bits/stdc++.h>

using namespace std;

struct Node

{

    int val;

    Node\* ne;

};

Node\* reverse(Node\* head)

{

    if(!head or !head->ne)

        return head;

    auto tail = head->ne;

    auto ans = reverse(head->ne);

    tail->ne = head;

    head->ne = NULL;

    return ans;

}

int main(int argc, char const \*argv[])

{

    auto head = new Node({1, new Node({2, new Node({3, new Node({4, NULL})})})});

    auto p = head;

    while(p)

    {

        cout << p->val << " ";

        p = p->ne;

    }

    puts("");

    head = reverse(head);

    p = head;

    while(p)

    {

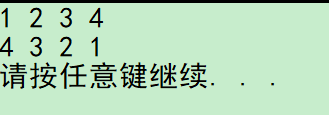
        cout << p->val << " ";

        p = p->ne;

    }

}

#### 测试结果



### 实验内容2: 假设二叉树采用二叉链表存储结点，设计递归算法判断两棵二叉树是否同构

#### 源代码

//2. 假设二叉树采用二叉链表存储结点，设计递归算法判断两棵二叉树是否同构

#include<bits/stdc++.h>

using namespace std;

template <class T>

struct Node

{

    T val;

    Node\* left, \*right;

};

template <class T>

class BiTree

{

public:

    Node<T> \*root;

public:

    BiTree(T val)

    {

        this->root = new Node<T>({val, NULL, NULL});

    }

    void bfs()

    {

        queue<Node<T>\*> q;

        q.push(root);

        Node<T>\* curLast = root, \*nextLast = NULL;

        cout << "Breath First Search Result:\n";

        while(q.size())

        {

            Node<T>\* t = q.front();

            q.pop();

            cout << t->val << " ";

            if(t->left)

                q.push(t->left), nextLast = t->left;

            if(t->right)

                q.push(t->right), nextLast = t->right;

            if(t == curLast)

                puts(""), curLast = nextLast;

        }

        puts("");

    }

};

bool dfs(Node<int>\* a, Node<int>\* b)

{

    int cntNull = (a == NULL) + (b == NULL);

    if(cntNull == 1)

        return false;

    else if(cntNull == 2)

        return true;

    if(a->val != b->val)

        return false;

    return dfs(a->left, b->left) and dfs(a->right, b->right);

}

bool judge(const BiTree<int>& tree1, const BiTree<int>& tree2)

{

    return dfs(tree1.root, tree2.root);

}

int main(int argc, char const \*argv[])

{

    BiTree<int> tree1(1);

    tree1.root->left = new Node<int>({2,

    new Node<int>({4, NULL, NULL}), NULL});

    tree1.root->right = new Node<int>({3,

    new Node<int>({6, NULL, NULL}), new Node<int>({7,

    new Node<int>({15, NULL, NULL}), NULL})});

    tree1.bfs();

    BiTree<int> tree2(1);

    tree2.root->left = new Node<int>({2,

    new Node<int>({4, NULL, NULL}), NULL});

    tree2.root->right = new Node<int>({3,

    new Node<int>({6, NULL, NULL}), new Node<int>({7,

    new Node<int>({15, NULL, NULL}), NULL})});

    tree2.bfs();

    auto ans = judge(tree1, tree2);

    if(ans)

        cout << "Tree1 = Tree2!!!";

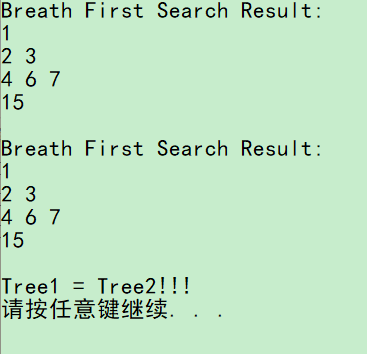
    else

        cout << "Tree1 != Tree2!!!";

    return 0;

}

#### 测试结果



### 实验内容3: 有n（n>3）个硬币，其中一枚是假币，由于假币的重量较轻，可以采用天平称重的方式找到这枚假币。请设计算法模拟寻找假币的过程

#### 源代码

//3. 有n（n>3）个硬币，其中一枚是假币，由于假币的重量较轻,

//可以采用天平称重的方式找到这枚假币。请设计算法模拟寻找假币的过程

#include<bits/stdc++.h>

using namespace std;

int findFakeCoin(vector<int>& coins, int l, int r)

{

    if(l == r)

        return l;

    if(l + 1 == r)

        return l < r ? l : r;

    int mid = (l + r) >> 1;

    double cntL = 0, cntR = 0;

    int lenL = mid - l, lenR = r - mid + 1;

    for(int i = 0; i < mid; i++)

        cntL += coins[i];

    for(int i = mid; i < r; i++)

        cntR += coins[i];

    if(cntL / lenL < cntR / lenR)

        return findFakeCoin(coins, l, mid);

    else

        return findFakeCoin(coins, mid, r);

}

int main(int argc, char const \*argv[])

{

    vector<int> coins{1, 1, 1, 1, 1, 0, 1};//ans is supposed to be 5

    cout << "Fake Coin index is " << findFakeCoin(coins, 0, coins.size() - 1);

    return 0;

}

#### 测试结果



### 实验内容4: 给定一个整数序列，每个元素出现的次数称为重数，重数最大的元素称为众数。请设计算法对一个递增有序的整数序列寻找众数

#### 源代码

//4. 给定一个整数序列，每个元素出现的次数称为重数，

//重数最大的元素称为众数。请设计算法对一个递增有序的整数序列寻找众数

#include<bits/stdc++.h>

using namespace std;

int main(int argc, char const \*argv[])

{

    vector<int> nums{1, 2, 2, 2, 3, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 7};

    int ans = 0, maxLen = 0;

    int dp = 1;

    for(int i = 1; i < nums.size(); i++)

    {

        dp = nums[i] == nums[i - 1] ? dp + 1 : 1;

        if(maxLen < dp)

            maxLen = dp, ans = nums[i];

    }

    cout << "Common Number is " << ans << endl;

    return 0;

}

#### 测试结果

