测试用例可能会输入很大的整型，这时候，用string处理才可以通过。这种题目特征比较明显，题目容易，看似简单，其陷阱在于大整数。

1023 判断乘二后是否是同一些数的排列。

#include <cstdio>

#include <string.h>

using namespace std;

int book[10];

int main() {

    char num[22];

    scanf("%s", &num);

    int flag = 0;

    int len = strlen(num);

    for(int i = len - 1; i >= 0; i--) {

        int temp = num[i] - '0';

        book[temp]++;

        temp = temp \* 2 + flag; //注意此处进位项flag的处理

        flag = 0;

        if(temp >= 10) {

            temp = temp - 10;

            flag = 1;

        }

        num[i] = (temp + '0');

        book[temp]--;

    }

    int flag1 = 0;

    for(int i = 0; i < 10; i++) {

        if(book[i] != 0)

            flag1 = 1;

    }

    if(flag == 1 || flag1 == 1) {

        printf("No\n");

    } else {

        printf("Yes\n");

    }

    if(flag == 1)

        printf("1");

    for(int i = 0; i < len; i++) {

        printf("%c", num[i]);

    }

    return 0;

}

1024 反转后相加判断是否是回文数

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

string s; //用string来存数

void add() {

    string t = s;

    reverse(t.begin(), t.end()); //这个string中的函数可以直接反转字符串

    int len = s.length(), carry = 0;

    for(int i = 0; i < len; i++) {

        s[i] = s[i] + t[i] + carry - '0';//进位的处理

        carry = 0;

        if(s[i] > '9') {

            s[i] = s[i] - 10;//进位的处理，直接剪去10

            carry = 1;

        }

    }

    if(carry) {reverse(s.begin(), s.end());s += '1';} //如果相加有进位

    reverse(s.begin(), s.end());

}

int main() {

    int cnt;

    cin >> s >> cnt;

    for(int i = 0; i <= cnt; i++) {

        string t = s;

        reverse(t.begin(), t.end());

        if(s == t) {

            cout << s << endl << i;

            return 0;

        }

        if(i != cnt) add();

    }

    cout << s << endl << cnt;

    return 0;

}