Leetcode234这道题的代码改得没有任务错误

链表要先实例化一个空指针，不用申请内存，它的next指向head，这个样子好处理一些，

Create链表的时候，p1负责申请内存，被赋值，p2负责把链表节点连起来并移动自己。

#include <stdio.h>

#include <iostream>

using namespace std;

struct ListNode {

int val;

ListNode \*next;

ListNode(int x) :val(x), next(NULL) {};

};

ListNode \*ReverseList(ListNode \*head);

ListNode \*Create();

void PrintList(ListNode \*head);

void main() {

ListNode \*head;

ListNode \*phead = (ListNode\*)malloc(sizeof(struct ListNode));

head = Create();

phead->next = head;

// PrintList(head);

ListNode \*slow, \*fast;

ListNode \*medium;

slow = fast = head;

while (fast != NULL && fast->next != NULL) {

slow = slow->next;

fast = fast->next->next;

}

//奇数链表

if (fast != NULL) {

medium = ReverseList(slow->next);

}

//偶数链表

else {

medium = ReverseList(slow);

}

bool result;

PrintList(medium);

while (medium) {

if (medium->val != head->val)

result = false;

medium = medium->next;

head = head->next;

}

result = true;

cout << endl;

cout << result;

// head = ReverseList(head);

//PrintList(head);

}

ListNode \*ReverseList(ListNode \*head) {

ListNode \*before = head->next;

ListNode \*after = head;

int count = 0;

while (before != NULL) {

count++;

head = before;

// if (before->next != NULL) {

before = before->next;

// }

// else {//head指针指到链表最后一个节点的时候，把head->next赋值为after，然后break

// head->next = after;

//if (count == 1)//把头节点的next置空

// after->next = NULL;

// break;

// }

//head = p->next;

head->next = after;

if (count == 1)//把头节点的next置空

after->next = NULL;

after = head;

// if (before->next == NULL)

// break;

}

return head;

}

//right create and print,important!!!!!!!!!!!!!!

//这个create必须用p2,因为p2才能把链表链接起来，只用p1的话p1每次p1=p1->next,但是p1之后会马上申请一个内存空间，就是新的内存位置，和之前的并没有连起来。

ListNode \*Create() {

ListNode \*head;

ListNode \*p1,\*p2;

head = NULL;

int val;

int count = 0;

p1 = (ListNode\*)malloc(sizeof(struct ListNode));

scanf("%d",&val);

p1->val = val;

while (p1->val != 0) {

count++;

if (count == 1) {

head = p1;

p2 = p1;

}

else {

p2->next = p1;

p2 = p1;

}

p1 = (ListNode\*)malloc(sizeof(struct ListNode));

scanf("%d",&p1->val);

}

p2->next = NULL;

return head;

}

void PrintList(ListNode \*head) {

ListNode \*p;

p = head;

while (p) {

cout << p->val;

p = p->next;

}

}