剑指18 删除链表的节点

//链表中删除一个节点

SListNode\* ps:待删除的链表

int add：待删除的位置

SListNode\* delete\_elem(SListNode\* ps,int add)

{

SListNode\* temp = ps;

//找到要删除的上一个节点

for(int i=1;i<add;i++)

{

if(temp==NULL)

{

printf("插入的位置无效\n");

return p;

}

temp=temp->next;

}

//指向待删除的节点

SListNode\* del=temp->next;

temp->next=temp->next->next;

free(del);

return ps;

}

剑指52 两个链表的第一个公共节点

//两个链表的第一个公共节点

struct Node\* find\_com\_node(Node\* p1,Node\* p2)

{

int len1=0;

int len2=0;

int len\_dif=0;

Node\* temp1=p1;

Node\* temp2=p2;

if(NULL==p1||NULL==p2)

{

}

while(temp1!=NULL)

{

len1++;

temp1=temp1->next;

}

while(temp2!=NULL)

{

len2++;

temp2=temp2->next;

}

temp1=p1;

temp2=p2;

if(len1>len2)

{

len\_dif=len1-len2;

for(i=0;i<dif;i++)

{

temp1=temp1->next;

}

}

else

{

len\_dif=len2-len1;

for(i=0;i<dif;i++)

{

temp2=temp2->next;

}

}

while(temp1!=NULL&&temp2!=NULL)

{

if(temp1==temp2)

{

return temp1;

}

temp1++;

temp2++;

}

}

返回倒数第 k 个节点

int kthToLast(ListNode\* head, int k) {

ListNode\* slow=head, \*fast=head;

while(--k)

fast=fast->next;

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

slow=slow->next;

fast=fast->next;

}

return slow->val;

}

反转字符串

void reverseString(vector<char>& s) {

int left=0;

int right=s.size()-1;

while(left<right)

{

// swap (nums[left],nums[right])

char temp = s[left];

s[left]=s[right];

s[right]=temp;

left++;

right--;

}

}

合并两个排序的链表

ListNode\* mergeTwoLists(ListNode\* l1, ListNode\* l2) {

if(l1==NULL) return l2;

if(l2==NULL) return l1;

if(l1->val<l2->val){

l1->next=mergeTwoLists(l1->next,l2);

return l1;

}

else{

l2->next=mergeTwoLists(l1,l2->next);

return l2;

}

}

### 剑指 Offer 50. 第一个只出现一次的字符

char firstUniqChar(string s)

{

if(s==" ")

return ' ';

unordered\_map<char,int> m;

for(int i=0;i<s.size();i++){

m[s[i]]++;

}

for(int i=0;i<s.size();i++){

if(m[s[i]]==1)

return s[i];

}

return ' ';

}