**Lab 5**

**Solve these problems for the linked list.**

class Solution {

public:

void deleteNode(ListNode\* node) {

ListNode \*temp = node->next;

node->val = node->next->val;

node->next = node->next->next;

delete temp;

}

};

class Solution {

public:

ListNode\* reverseList(ListNode\* head) {

if(head == NULL) return NULL;

ListNode\* current = head;

ListNode\* prev;

ListNode\* next = NULL;

while(true){

prev = current->next;

current->next = next;

next = current;

if(prev != NULL)

current = prev;

else

break;

}

return current;

};

class Solution {

public:

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

if(l1==NULL && l2==NULL)

return NULL;

else if(l1==NULL)

{

return l2;

}

else if(l2==NULL)

{

return l1;

}

else

{

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

{

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

return l1;

}

else

{

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

return l2;

}

}

return NULL;

}

};

bool hasCycle(ListNode \*head) {

if (!head) return false;

unordered\_map<ListNode\*, int> map;

while(head!=NULL){

if(map.count(head) > 0){

return true;

}else{

map[head] = 1;

}

head = head->next;

}

return false;}

ListNode\* partition(ListNode\* head, int x) {

ListNode\* cur = new ListNode(0);

cur->next = head;

head = cur;

while (cur->next && cur->next->val < x)

cur = cur->next;

ListNode\* insPos = cur;

ListNode\* tmp;

while (cur->next) {

if (cur->next->val < x) {

tmp = cur->next;

cur->next = tmp->next;

tmp->next = insPos->next;

insPos->next = tmp;

insPos = insPos->next;

} else {

cur = cur->next;

}

}

return head->next;

}

**7.**

class Solution {

public:

int calc(ListNode\* head,int k,int len){

ListNode\* temp = head;

if(head->next==NULL && k==1) return head->val;

int f = len-k+1;

while(--f>0){

temp=temp->next;

//f-=1;

}

return temp->val;

}

bool isPalindrome(ListNode\* head) {

if(head==NULL || head->next==NULL) return true;

ListNode\* temp = head;

int len = 1;

while(temp->next!=NULL){

len+=1;

temp= temp->next;

}

int n = len/2;

stack<int> s;

while(n > 0){

s.push(head->val);

head=head->next;

n-=1;

}

//cout<<len;

if(len%2==0){

while(head!=NULL){

if(head->val == s.top()) {

s.pop();

head=head->next;

}else return false;

}

return true;

}

else {

head=head->next;

while(head!=NULL){

if(head->val == s.top()) {

s.pop();

head=head->next;

}else return false;

}

return true;

}

}

};

**8,**

class Solution {

public:

ListNode\* sortList(ListNode\* head) {

for(ListNode\* i = head; i != NULL; i = i->next){

for(ListNode\* j = i->next; j != NULL; j = j->next){

if(i->val > j->val){

swap(i->val, j->val);

}

}

}

return head;

}

};