**Lab 9 Part I 链表**

# **Problem 1.** 实现链表类LinkedList。节点类ListNode定义和链表类LinkedList声明已给出，需实现插入节点函数、删除节点函数、创建链表函数、查找函数、打印链表函数、反转链表函数、删除链表倒数第 N个节点函数、查找链表中间节点、链表升序排序函数、删除排序链表中的重复元素函数。

class ListNode

{

public:

ListNode() : val(0), next(0) {}

ListNode(int x) : val(x), next(0) {} int val;

ListNode \*next;

}; //节点类定义

class LinkedList

{

public:

LinkedList(); //默认构造函数，创建head节点

~LinkedList(); //析构函数，删除所有节点，包括head节点

void insert(int position, int element); //向链表指定位置前添加新的节点void remove(int position); //移除链表指定位置的节点

void createList();//从键盘上读入若干个int数，基于这些数创建一个链表int find(int position); //查找指定位置的结点，返回节点的值

void print(); //打印链表，例：[1,2,3,4]：1 2 3 4

void reverseList(); //反转链表，例：[1,2,3,4]->[4,3,2,1]

void removeNthFromEnd(int n); //删除链表的倒数第N个结点，例：N=2，[1,2,3,4]-> [1,2,4]，要求只能遍历一次链表

int findMiddle();//查找链表的中间节点，返回节点的值，要求只能遍历一次链表

void sortList(); //链表升序排序，例：[5,4,9,1]->[1,4,5,9]，注：只允许在链表上进行操作，不允许将数据读到数组里进行排序

void deleteDuplicates(); //删除排序链表中的重复元素，例：[1,1,1,2,3]->[1,2,3] private:

ListNode \*head;//指向链表头节点

}; //链表类

#include<iostream>

using namespace std;

class ListNode {

public:

int val;

ListNode\* next;

ListNode():val(0), next(0){}

ListNode(int x):val(x), next(nullptr){}

};//节点类定义

class LinkedList {

public:

ListNode\* head;

int nodeNum;

LinkedList() {

head = new ListNode();

nodeNum = 0;

}

~LinkedList() {

for (int i = nodeNum; i >=1; i--)

remove(i);

delete head;

}

void insert(int position, int element) {

if (position < 1 || position > nodeNum + 1)

return;

ListNode\* newNode = new ListNode(element);

ListNode\* p = head;

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

p = p->next;

newNode->next = p->next;

p->next = newNode;

nodeNum++;

}

void remove(int position) {

if (position < 1 || position > nodeNum)

return;

ListNode\* p = head;

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

p = p->next;

ListNode\* q = p->next;

p->next = q->next;

delete q;

nodeNum--;

}

void createList(){

int data;

int position = 1;

while (1) {

cin >> data;

insert(position++, data);

}

}

int find(int position) {

if (position < 1 || position > nodeNum)

return 0;

ListNode\* p = head;

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

p = p->next;

return p->val;

}

void print() {

ListNode\* p = head;

while (p) {

p = p->next;

if (p != nullptr)

cout << p->val << " ";

}

cout << endl;

}

void reverseList(ListNode\* &head) {

if (head == nullptr || head->next == nullptr)

return;

ListNode\* beg = nullptr;

ListNode\* mid = head;

ListNode\* end = head->next;

while (1) {

mid->next = beg;

if (end == nullptr)

break;

beg = mid;

mid = end;

end = end->next;

}

head = mid;//ListNode\* reverseList() {

//return head;

}

void removeNthFromEnd(int n) {

ListNode\* fast = head;

ListNode\* slow = head;

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

fast = fast->next;

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

fast = fast->next;

slow = slow->next;

}

ListNode\* q = slow->next;

slow->next = q->next;

delete slow;

}

int findMiddle() {

ListNode\* fast = head;

ListNode\* slow = head;

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

fast = fast->next->next;

slow = slow->next;

}//while(fast->next != nullptr && fast->next->next != nullptr)

return slow->val;

}

void deleteDuplicates(ListNode\* head) {

ListNode\* dummyhead = new ListNode(0);

dummyhead->next = head;

ListNode\* prev = dummyhead;

ListNode\* cur = prev->next;

while (cur != nullptr) {

if (prev->val != cur->val)

prev = prev->next;

else

prev->next = cur->next;

cur = cur->next;

}

head = dummyhead->next;

}

void sortList() {

ListNode\* dummy = new ListNode(0);

for (ListNode\* p = head; p;) {

ListNode\* cur = dummy;

ListNode\* next = p->next;

while (cur->next && cur->next->val <= p->val)

cur = cur->next;

p->next = cur->next;

cur->next = p;

p = next;

}

head = dummy->next;

}

};

# **Problem 2.** 基于Problem 1定义的LinkedList类，创建两个有序的链表l1和l2，编写函数，将这两个有序的单链表合并成一个有序链表。

//我的思路：用find找出所有节点储存到同一个数组，数组比较大小，再用insert合并一个有序链表

**Problem 3.** 基于Problem 1定义的ListNode和LinkedList类，编写函数创建一个带环的链表（将链表的最后一个节点的next指向中间的某个节点）；编写另外一个函数，检查任意给定的一个链表是否有环。

bool loop(Node\* head) {

Node\* fast = head;

Node\* slow = head;

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

fast = fast->next->next;

slow = slow->next;

if (fast == slow)

return true;

}

return false;

}

**Problem 4.** 给定下面的ListNode节点类，注意节点里面有两个指针next和random；编写函数，创建一个复杂链表，其中节点的next指针指向后面一个节点，random指针随机指向链表中的一个节点或者是NULL；编写另外一个函数，复制所创建的复杂链表（即创建另外一个复杂链表，其中节点之间的连 接关系和给定的复杂链表一模一样）。

//复杂链表的复制

class Solution {

public:

Node\* copyRandomList(Node\* head) {

Node\* cur = head;

while (cur) {

Node\* copyNode = new Node(cur->val);

Node\* next = cur->next;

cur->next = copyNode;

copyNode->next = next;

cur = next;

}

Node\* cur = head;

while (cur) {

Node\* copyNode = cur->next;

if (cur->random == nullptr)

copyNode->random = nullptr;

else

copyNode->random = cur->random->next;

cur = copyNode->next;

}

cur = head;

Node\* copyhead = new Node(0);

Node\* ans = copyhead;

while (cur) {

Node\* copyNode = cur->next;

Node\* next = copyNode->next;

cur->next = next;

copyhead->next = copyNode;

copyhead = copyhead->next;

cur = next;

}

return ans->next;

}

};

class ListNode

{

public:

int val; ListNode \*next;

ListNode \*random;

}; //节点类定义