#include<iostream>

using namespace std;

//节点类

class Node {

public: int data;

Node\* next;

Node(int n) {

data = n;

next = NULL;

}

Node() {};

};

//链表一

class list1 {

public:

Node\* front;

int count;

list1() {

front = NULL;

count = 0;

}

//删除节点

void Delete(int n) {

if (front == NULL) { return; }//空链表返回

//只有一个头节点时

if (front->data == n && front->next == NULL) {

front = NULL;

count--;

return;

}

while (front->data == n) {

if (front->data == n && front->next != NULL) {

front = front->next;

count--;

}

if (front->data == n && front->next == NULL) {

front = NULL;

count--;

return;

}

if (front == NULL)return;

}

Node\* m = new Node();

Node\* p = front;//p一直在m前面，这样好删除

m = p->next;

while (true) {

if (m == NULL)return;//到末尾了就结束

//末尾情况

if (m->data == n && m->next == NULL) {

p->next = NULL;

count--;

return;

}

//中间情况

else {

if (m->data == n) {

p->next = m->next;

count--;

}

m = m->next;

p = p->next;

if (m == NULL)return;//到末尾了就结束

}

}

}

//插入函数

void insert(int n) {

Node\* p = new Node(n);

if (front == NULL) {

front = p;

count++;

}

else {

Node\* temp = front;//用来找尾节点

while (temp->next != NULL) {

temp = temp->next;

}//找到尾节点

temp->next = p;

count++;

}

}

//逆置函数

void reverse() {

if (front == NULL || front->next == NULL) { return; }//空链表或者只有一个

//利用三个指针

Node\* x = new Node();

Node\* y = new Node();

Node\* z = new Node();

x = front->next;//防止失去联系

front->next = NULL;//将头节点孤立

y = x->next;

x->next = front;

while (1) {

z = x;

x = y;

if (x == NULL)break;

y = x->next;

x->next = z;

}

front = z;

}

};

//链表二

class list2 {

public:

Node\* front;

Node\* tail;

list2() {

front = tail = NULL;

}

//逆置函数

void reverse() {

if (front == tail) { return; }

//利用两个指针

Node\* cur1 = new Node();

Node\* cur2 = new Node();

cur1 = front->next;

front->next = NULL;//将头节点孤立

tail->next = front;

cur2 = cur1->next;

while (cur1 != tail) {

cur1->next = tail->next;

tail->next = cur1;

cur1 = cur2;

cur2 = cur1->next;

}

tail = front;

front = cur1;

tail->next = front;//因为是循环链表，所以逆置后也不要忘记这步

}

//插入函数

void insert(int n) {

Node\* p = new Node(n);

if (front == NULL && tail == NULL) {

front = tail = p;

tail->next = front;

}

else {

tail->next = p;

tail = p;

tail->next = front;

}

}

};

int main() {

list1 A;

list2 B;

int n;

cout << "请输入链表A的数据：";

while (cin >> n) {

A.insert(n);

}

cin.clear();//将输入流恢复正常

cout << "请输入链表B的数据：";

while (cin >> n) {

B.insert(n);

}

A.reverse();

B.reverse();

Node\* i = new Node();

Node\* j = new Node();

i = A.front;

cout << "链表A的数据: ";

for (; i != NULL;) {

cout << i->data << " ";

i = i->next;

}

cout << endl;

cout << "链表B的数据: ";

i = B.front;

for (; ;) {

cout << i->data << " ";

i = i->next;

if (i == B.front)break;

}

cout << endl;

j = B.front;

A.Delete(j->data);

//查找与B中数据相同的节点

j = B.front->next;

while (j != B.front) {

A.Delete(j->data);

j = j->next;

}

i = A.front;

cout << "删除与B数据相同节点后的A： ";

for (; i != NULL;) {

cout << i->data << " ";

i = i->next;

}

cout << endl;

int k;

cin.clear();

cout << "请输入k: ";

cin >> k;

if (k <= 0 || k > A.count) { cout << "k值超过范围 "; }

else {

cout << "A中倒数第k个节点为： ";

i = A.front;

j = A.front;

for (int m = 0; m < k - 1; m++) {

j = j->next;

}//先让j领先i k步

for (; j->next!=NULL; ) {

j = j->next;

i = i->next;

}//当j到尽头，i指向倒数第k个节点

cout << i->data << endl;

}

system("pause");

return 0;

}