# 1、单链表中结点按要求调序1

设带头结点的正整数单链表包含n个元素结点，即原单链表可以表示为：a0-> a1-> ...-> am-> ...-> an-2-> an-1。 设计算法，重新调整单链表中的结点次序。要求：

将原链表的前k个元素和最后k个元素调整为 a0 -> an-1 -> a1 -> an-2-> a2 ->an-3-> ......->ak-1->an-k->（剩余元素维持原状）

void re\_arrange1(int k) {

Node\* q, \* p,\*p1,\*q1;

Node\* r, \* midt;

q = p = head;

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

p = p->next;

while (p->next) {

p = p->next;

q = q->next;

}

midt = q;

p = q->next;

midt->next = NULL;

while (p) {

r = p->next;

p->next = q->next;

q->next = p;

p = r;

}

q = q->next;

p = head->next;

midt->next = NULL;

r = head;

while (q) {

p1 = p->next;

q1 = q->next;

r->next = p;

p->next = q;

r = q;

p = p1;

q = q1;

}

r->next = p;

}

# 2、单链表中结点按要求调序2

设带头结点的正整数单链表包含n个元素结点，即原单链表可以表示为：a0-> a1-> ...-> am-> ...-> an-2-> an-1。 设计算法，重新调整单链表中的结点次序。调整原则：

指定位置m，将原链表调整为 a0 -> am+1 -> a1 -> am+2 -> .... （无配对元素保持原序）

void re\_arrange2(int k) {

Node\* q, \* p, \* p1, \* q1;

Node\* r, \* midt;

q = p = head;

for (int i = 0; p && i <= k; i++)

p = p->next;

if (!p) return;

q = p->next;

p->next = NULL;

p = head->next;

r = head;

while (p && q) {

p1 = p->next;

q1 = q->next;

r->next = p;

p->next = q;

r = q;

p = p1;

q = q1;

}

if (p) r->next = p;

else

r->next = q;

}

# 3、单链表中结点按要求调序3

【问题描述】

设带头结点的正整数单链表包含n个元素结点，即原单链表可以表示为：a0-> a1-> ...-> am-> ...-> an-2-> an-1。 设计算法，重新调整单链表中的结点次序。调整原则有：

指定关键字key，将原链表中值为key的结点移动到链表尾部，其他结点的相对位置不变。例如：链表原结点为3->10->3->3->5->6，key为3，则调整后链表为10->5->6->3->3->3。

void re\_arrange3(int key) {

Node\* q, \* p;

p = head->next;

q = p;

while (p) {

if (p != q && p->entry != key) {

q->entry = p->entry;

p->entry = key;

p = p->next;

q = q->next;

}

else

if (p->entry != key) {

p = p->next;

q = q->next;

}

else

p = p->next;

}

}