P184 5

//合并链表

class chainList;

class chainNode

{

friend class chainList;

private:

int data;

chainNode\* next;

};

class chainList

{

private:

chainNode\* first;

public:

chainList combine(chainList x, chainList y);

};

//合并函数

chainList chainList::combine(chainList x, chainList y)

{

//创建用于表示结果的链表

chainList z = new chainList;

//判断x y链表是否有一个为空

if(x.first==NULL)

{

z.first = y.first;

}

if(y.first==NULL)

{

z.first = x.first;

}

chainNode\* xptr = x.first;

chainNode\* yptr = y.first;

chainNode\* zptr = z.first;

//当两个链表都没读取完

while(xptr&&yptr)

{

if(xptr->data<=yptr->data)

{

//判断z链表是否为空

if(zptr==NULL)

{

zptr = xptr;

z.first = xptr;

xptr = xptr->next;

}

else

{

zptr->next = xptr;

xptr = xptr->next;

zptr = zptr->next;

}

}

else

{

//判断z链表是否为空

if(zptr==NULL)

{

zptr = yptr;

z.first = yptr;

yptr = yptr->next;

}

else

{

zptr->next = yptr;

yptr = yptr->next;

zptr = zptr->next;

}

}

}

if(xptr==NULL)

{

zptr->next = yptr;

}

else

{

zptr->next = xptr;

}

return z;

}

P184 6

//反转链表，双向遍历

class chain;

class chainNode

{

friend class chain;

private:

int data;

chainNode\* next;

};

class chain

{

private:

chainNode\* first;

public:

void moveLeft(chainNode\* l,int n);

void moveRight(chainNode\* r,int n);

};

//向左移动的函数

void chain::moveLeft(chainNode\* l,chainNode\* r,int n)

{

//判断n是否合法

if(n<0)return;

//向左遍历

chainNode\* tmp;

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

{

tmp=l;

l=l->next;

tmp->next = r;

r = tmp;

//判断是否到达最左端

if(l->next==NULL)

{

l->data = 0;

r=l;

break;

}

}

}

//向右移动的函数

void chain::moveRight(chainNode\* l,chainNode\* r,int n)

{

//判断n是否合法

if(n<0)return;

//向右遍历

chainNode\* tmp;

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

{

tmp=r;

r=r->next;

tmp->next = l;

l = tmp;

//判断是否到达最右端

if(r->next==NULL)

{

r->data = 0;

l=r;

break;

}

}

}

P194 4

//设x1，x2...xn是一个链表中的元素，xi是一个整数。编写一个C++函数，计算表达式Sum（xi\*（xi+5））

class chainList;

class chainNode

{

friend class chainList;

private:

int data;

chainNode\* next;

};

class chainList

{

private:

chainNode\* first;

public:

int mySum(chainList\* list);

};

//求和函数

int chainList::mySum(chainList\* list)

{

//判断是否是空的链表

if(list->first==NULL)

return 0;

//判断元素是否有五个

chainNode\* lptr = list->first;

chainNode\* rptr = list->first;

for(int i=1;i<=5;i++)

{

rptr = rptr->next;

if(rptr==NULL)

return 0;

}

//满足计算的要求，开始求和

int sum = 0;

while(rptr!=NULL)

{

sum+=lptr->data\*rptr->data;

lptr = lptr->next;

rptr = rptr->next;

}

return sum;

}