1. 将两个递增的有序链表合并为一个递增的有序链表。要求结果链表仍使用原来两个链表的存储空间，不另外占用其他的存储空间。表中不允许有重复的数据。

void Mergelist\_L(Linklist &La, Linklist &Lb, Linklist &Lc)

{

pa = La->next;

pb = Lb->next;

Lc = pc = pa;

while (pa && pb)

{

if (pa->data > pb->data)

{

pc->next = pb;

pc = pb;

pb = pb->next;

}

else if (pa->data < pb->data)

{

pc - next = pa;

pc = pa;

pa = pa->next;

}

else

{

pc->next = pa;

pc = pa;

pa = pa->next;

q = pb->next;

delete pb;

pb = q;

}

}

pa->next = pa ? pa : pb;

delete Lb;

}

1. 将两个非递减的有序链表合并为一个非递增的有序链表。要求结果链表仍使用原来两个链表的存储空间，不另外占用其他的存储空间。表中允许有重复的数据。

void MergeList\_L(Linklist &La, Linklist &Lb,Linklist &Lc)

{

pa = La->next;

pb = Lb->next;

Lc = pc = La;

Lc->next = NULL;

while (pa || pb)

{

if (!pa)

{

q = pb;

pa = pa->next;

}

else if (!pb)

{

q = pa;

pb = pb->next;

}

else if (pa->data >= pb->data)

{

q = pb;

pb = pb->next;

}

else

{

q = pa;

pa = pa->next;

}

q->next = Lc->next;

Lc->next = q;

}

1. 设计一个算法，通过一趟遍历在单链表中确定值最大的结点

void Maximum\_L(Linklist &La)

{

pa = La->next;

pmax = pa;

while(pa->next != NULL)

{

pa = pa->next;

if (pmax->data < pa->data)

{

pmax = pa;

}

}

return pmax->data

}

1. 设计一个算法，通过一趟遍历，将链表中所有结点的链接方向逆转，且仍利用原表的存储空间。

void reverse\_L(Linklist &La)

{

p = L->next;

l->next = NULL;

while (p != NULL)

{

q = p->next;

p->next = L->next;

L->next = p;

p = q;

}

}