（1）、单链表就地逆置

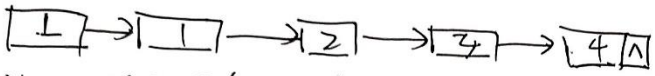
代码：

单链表实现就地逆置

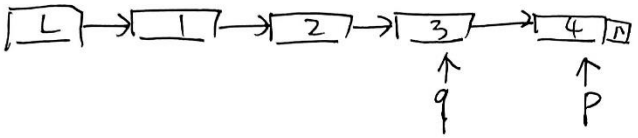
```
//找到单链表L最后一个元素的指针
int * ListEnd(LinkList L)
{
  LinkList p;
  p=first;
  while(p)
  {
      p=p→next;
  }
  return p;
}

//查找单链表L的长度
int  ListLength(LinkList L)
{
    LinkList p;
    p=first;
    int count=0;
    while(p)
    {
        p=p→next;          count++;
    }
    return count;
}

//实现对单链表就地逆置
void ListReverse{
  int n;
  //算出L的长度
  n = LinkList(L);
  //使p指向最后一个节点
  p = ListEnd(L);
  //q是p的前驱节点
  q -> next = p;
  //进行n-1次循环
  for(int i = 0; i < n-1; i++){
    p -> next = L - > next;
    L -> next = q -> next;
    //将q的指针域置空
    q -> next = null;
    //p指向q所指的指针
    p = q;
    //q所指的节点再次变成p的前驱节点
    q -> next = p;
  }
}
```

图解：

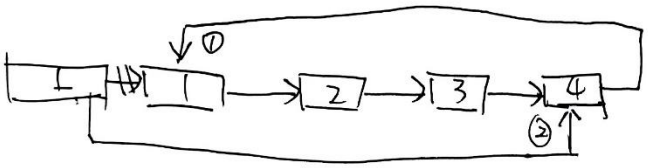
①计算L的长度（假设长度为4）.



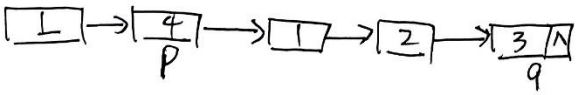
②使P指针指向最后一个结点，而q为P的前驱结点



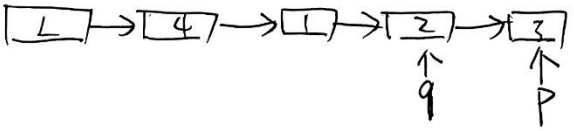
③ p→next = L→next ; L→next = q→next ; q→next = null



即



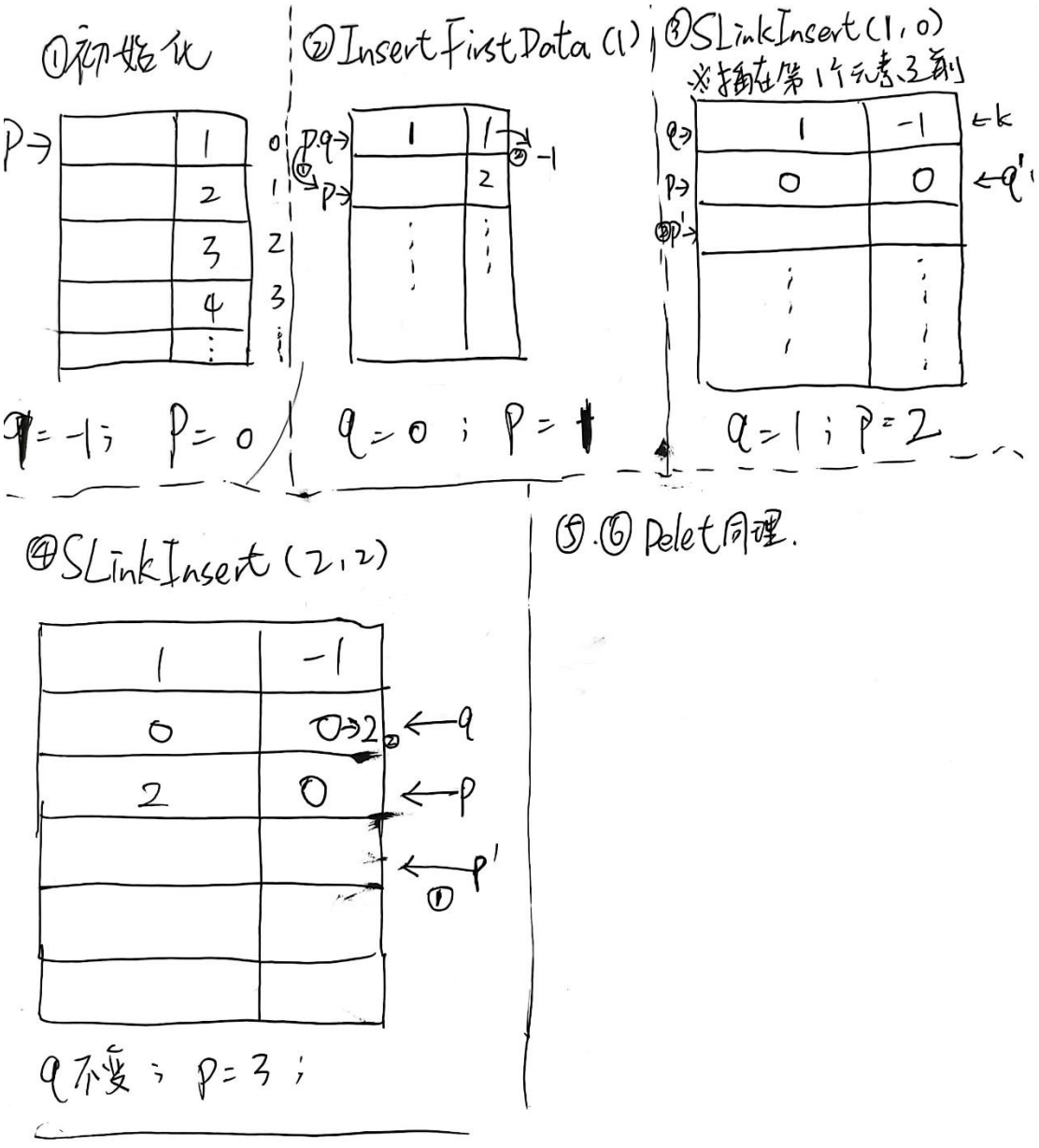
④ p=q ; q→next = p ;



注：只需循环 n-1 次

2、不带头结点静态链表的插入和删除算法（完整代码有点长，pdf 放不下。在压缩包下有相关的图片和源码（SLinkListInsertAndDelet））

主要思想：p、q 分别指向备用链表和链表的第一个元素，初始值备用链表就是空链表 p = 0，而链表不存在，q = -1；插入元素之前链表内需要一个元素，故定义了一个 InsertFirstData 函数。除此之外插入和删除元素时需要讨论删除元素的位置（若为第一个元素则需要特殊考虑）



3、双链表删除最大值节点

代码：

HW 3-3

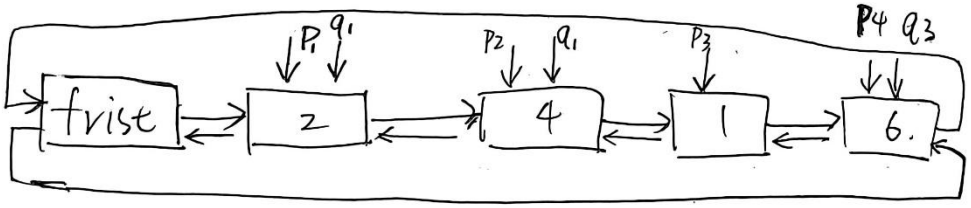
```
//求双链表表长
int DllLength()
{
  int cnt = 0;
  int m = frist;
  while(m -> next != frist)
  {
    m = m -> next;
    cnt ++;
  }
  return cnt;
}
//比较，若p的data大于q的data则将p赋值给q
void compare(int* p, int* q)
{
  if(p -> data > q -> data) q = p;
}
//删除双链表最大值节点
void DeleteDllMax(Dll)
{
  //p、q初始都指向第一个首元节点
  int p = first -> next, q = first -> next;
  for(int i = 0; i < DllLength(); i++)
  {
    //p每移动一次就要将p和q的data进行对比
    p = p -> next;
    compare(p, q);
  }
  //遍历完毕后删除q（指向最大值）
  q -> prior -> next = q -> next;
  q -> next -> prior = q -> prior;
  free(q);//释放q节点
  return;
}
```

图解：

①创建两个指点 p、q，p 用于遍历链表所有元素 q 用于指向当前最大的元素的数组。

② p每经过一次节点就要对较 P.data 和 q.data，若 p.data > q.data 则q指向p。

③当 P、next = first 时遍历完毕，查看此时q所指元素，使 q→prior→next = q→next； q→next→prior = q→prior free(q)即可。



①②③后：