1. 试写出顺序存储结构下逆转线性表的算法，要求使用最小的附加空间。

思路：第一个和最后一个互换，第二个和倒数第二个，一直到中间的两个或者一个。即 for i:=头 to ((头+尾) div 2) do swap(a[i],a[尾-i+头]) 空间代价是一个元素，时间代价是 O(n)。

典型错误：很多人直接写 swap(a[i],a[尾-i+头])，思路是对的，但最好要写出具体的代码；且在 for 循环中，有的同学不知道条件应该为 **i< n/2**,n 为线性表长度。

参考代码：

```
template <typename DataType>
const int N = 1024;
struct list{
DataType data[N];
int max;
};
typedef struct list List;
void reverseList(List &l)
{
for(int i=0;i<max/2;++i)
{
l.data[i] = l.data[i]+l.data[max-i-1];
l.data[max-i-1] = l.data[i]-l.data[max-i-1];
l.data[i] = = l.data[i]-l.data[max-i-1];
}
}
```

2. 从顺序存储结构线性表 a 中删除第 i 个元素起的 k 个元素。

思路 :把从第 i+k 个元素起的元素前移到第 i 个元素起的后续位置上。

典型错误：对于开始 i，k 参数如有输入错误的情况没有考虑，或者

没有考虑全，本题条件应该是 i<1||k<0||i+k-1>a.length。

对于循环结束的条件关系没有理清，对于从第 i+k 个元素起的元素移到 ,第 i 个元素起的后续位置的正确表述为 for(i=i+k-1;j<a.length;j++) A[j-k]=A[j] //注意红色部分易错;

最后应当注意的是删除元素后，线性表的长度发生了变化，不少同学都漏写了 a.length-=k;

参考代码：

```
Status DeleteK(SqList &a,int i,int k)//删除线性表 a 中第 i 个元素起的 k 个元素
{

if(i<1||k<0||i+k-1>a.length) return INFEASIBLE;

for(count=1;i+count-1<=a.length-k;count++) //注意循环结束的条件

a.elem[i+count-1]=a.elem[i+count+k-1];

a.length-=k;

return OK;

}//DeleteK
```

3．已经线性表中的元素非递减有序排列，并以带头结点的单链表做存储结构，试写一高效算法，删除表中所有值相同的多余元素。

思路：p 指向当前结点，q 指向下一结点，若相等删除 q，并把 q 指向下一个，继续操作 。

典型错误：定义了指针，但是对指针的初始指向含糊不清，或者没有指向；

对于单链表的指向下一个结点，不要与顺序存储混淆，如 q 指向下个结点，不能为 q++；

对于删除重复的元素代码为：

q=p; p=p->next; prev->next=p; free(q); 若删除 P 指向的，则要先将 q 指向 p，p 指向后继，最后释放的是 q；free（q）代码很多人没有写

参考代码：

```
Status DeleteEqual_L(&L)
{
 //删除有序链表中的重复元素
 if (p=L->next)
 {
  prev=p;
  p=p->next;
 }
 while (p)
 {
  if (p->date==prev->date)
  {
   q=p;
   p=p->next;
   prev->next=p;
   free(q);
  }
  else
  {
   prev=p;
   p=p->next;
  }
 }//while
 return OK;
}//DeleteEqual_L
```

4.写出逆置线性单链表的算法

思路：将工作指针指向第一个元素结点，然后将头结点指针域置空，接着将链表各结点从第一个开始直至最后一个，依次前插至头结点后，则逆置成功。

典型错误：思路不清，没考虑好逆置方法后而写代码，则代码也很乱，根本达不到逆置结果；主要是修改所有结点的指针将第一个结点指针域置空，其它结点指针域指针指向其前一个结点，最后将头结点指向最后一个结点；最后也要注意要将头结点指向最后一个结点，一定要全面考虑。其实这里可分两种情况，带头结点和不带头结点的情况，思路是一样的，只是算法上没有头结点的处理。代码还是比较简单的。

参考代码：

```
Linklist     reverse_list(    )
    {          p=head—>next;      head—>next=null;
       While(p!=null)
                    { r=p—>next;    p—>next=head—>next; head—>next=p; p=r}
          Return OK;
    }
```