专题四“查找和排序”课后练习：

1. 下面程序段用于求两个n\*n矩阵相乘的算法，试求其时间复杂度。

for(i=0;i<n;i++)

for(j=0;j<n;j++){

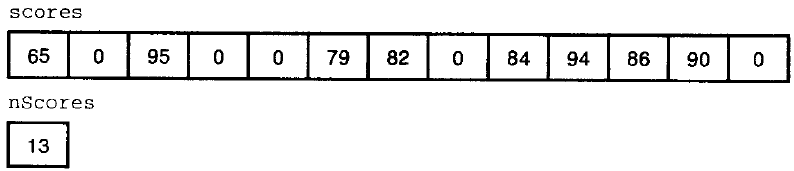
c[i][j]=0;

for(k=0;k<n;k++)

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

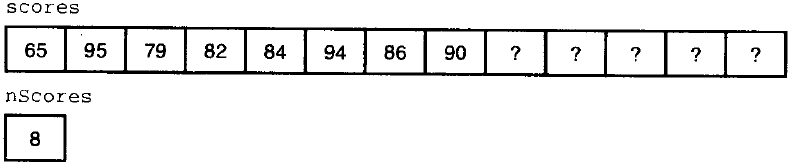
2.编写一个函数RemoveZeroElements(array，n)，检查整型数组的每个元素，删除任何值为0的元素。由于该操作将改变数组的有效长度，因此需要返回新数组的有效长度。 例如，假设scores是一个含有可选考试项目的分数的数组，nScores指出数组的有效长度，如下表所示：

****

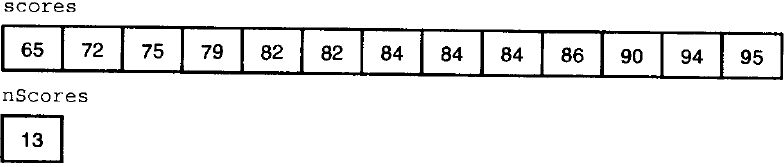
这里语句：

nScores = RemoveZeroElements(sCores, nScores);

应该删除其中所有为0的分数，把上表中的数组压缩成下表：



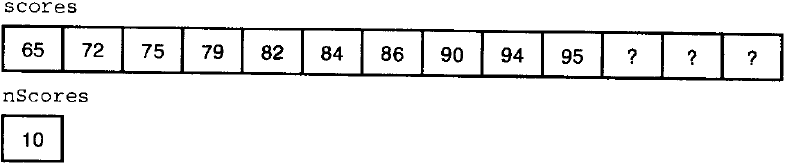
3. 编写一个函数RemoveDuplicates(array，n)，从某个已排好序 的整型数组中删除所有重复的数值，只保留其中的一个拷贝。该需要返回新数组的有效长度。 例如，假设数组scores中包含以下数值：



语句：

nScores = RemoveDuplicates(scores, nScores);

应该删除重复分数，把上表中的数组压缩成下表：

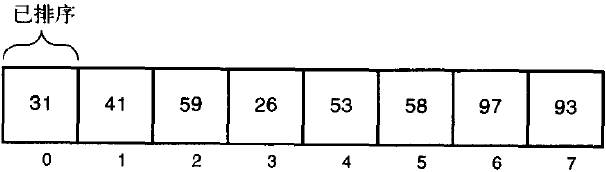
****

4. 很多算法问题在它们的解决方案中都和排序有关。例如，可以根据随机键值“排序”，以打乱一个数组。一种方法就是使用选择排序算法，然后把找到最小值的位置改成选择一个随机位置。该结果是一个打乱算法，其中各种可能的输出是等概率的。

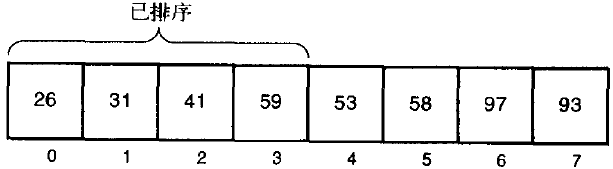
编写一个程序shuffle.c，通过随机排序输出1-52之间的整数。

5. 插入排序算法原理和选择排序一样，它也是一次检查数组的每一个元素。然而在此过程中，每一步的目的不是在剩余元素中找到最小值，把它与正确的位置值交换，而是确保检查过的元素是以正确的次序排好的。虽然随着新的元素的加入，这些元素的位置可能还要改变，但是插入的元素和以前的序列在一起，仍然构成按特定顺序排列的序列。

例如，考虑如下数据，插入排序的第一个循环周期不需要任何操作，因为一个元素的数组总是排好序的。



接下来的两个循环周期也不需要重新排列数组，因为31-41-59形成一个有序的子数组。第一次有意义的操作出现在下一个循环周期中，当需要将26插入到前面已经排列好的序列中时，为了找到26应该放的位置，要将前面已经排好序的元素向后移，寻找26的位置。在每一步中，为了给26准备存储位置，需要将其他元素移动一个位置，最终到达位置0。因此，第四个循环周期后得到的序列为：



在接下去的每一步中，需要做的是将当前元素插入到已经排好序的子数组中，直到到达数组末尾为止。

用插入排序法实现函数SortIntegerArray，并尝试分析该算法的时间复杂度。