基础：70 分

1. 算法设计（10 分）：设顺序表 va 中的数据元素递增有序。试写一算法，将 x 插入到顺序表的适当位置上，且保持表的有序性。

解：关键在于利用元素有序这个特点，寻找插入位置时，不必遍历整个链表。完整代码见附件。主要流程包括：寻找插入位置：

（一） 查找首个大于 x 的顺序表元素

```
for (i = 0; i < S->length; i++)
{
  if (S->elem[i] >= e) break;
}
if (S->elem[i] == e) return ERROR;
```

（二） 顺移元素，腾出插入位置。

```
for (j = S->length - 1; j >= i; j--)
{
  S->elem[j + 1] = S->elem[j];
}
S->elem[i] = e;
S->length++;
```

算法可以进一步优化，将比较和移位同时进行，即从表尾开始，每比较一次就移动一次（如果当前元素大于 x，则后移）。

2. 算法设计（10 分）：假设以两个元素值非递减有序排列的线性表 A 和 B 分别表示两个集合，在同一表（A 或 B）中可能存在值相同的元素，求 A 和 B 的交集，存放在 A 表空间中，要求新生成的交集 A 表中的元素值各不相同且按非递减有序排列。（本题线性表采用顺序存储映像）

解：要点在于：

第一：旧线性表 A 和新线性表 A 共用存储空间，利用不同的下标实现了这一点。完整代码见附件。

第二：充分利用线性表有序这一特点，控制比较的轮次。

算法设计思路如下：

/*k:指向新 A 表中当前元素位置，i:指向旧 A 表当前元素位置，j:指向 B 表当前元素位置*/

/*1.若 a[i]<b[j],i++，即继续用 A 表下一个元素比较*/

/*2.若 a[i]==b[j]。这是 A^B 元素，若不重复 a[k]!=a[i]，则入新 A 表，k++,a[k]=a[i]。此外，i++,j++，即继续比较 A,B 表下一个元素*/

/*3.若 a[i]>b[j]。j++，即继续用 B 表下一个元素比较*/

/*4.A，B 表任意一个元素都已被访问完毕时，交集计算完成，元素个数=k+1*/

代码：

```
while ((i < A->length) && (j < B->length))
{
  if (A->elem[i] < B->elem[j]) { i++; continue; }
  if (A->elem[i] > B->elem[j]) { j++; continue; }
  if (A->elem[i] == B->elem[j]) {
    if ((k != -1) || (A->elem[k] != A->elem[i]))
    {
      A->elem[++k] = A->elem[i];
    }
    i++; j++;
  }
}
A->length = k + 1;
```

3. 算法设计（10 分）：编程实现单链表基本操作，例如，创建结构，插入结点，删除结点等等。

解：完整代码见附件，注意不同操作时，循环终止条件的控制。例如，插入结点时，可以在链尾结点后插入新结点，所以指针可以移动到链尾结点。但删除结点时，需要目标结点的前驱结点指针。因此，要删除尾结点，至多向链尾方向移动到链尾结点的前一个结点（前驱）。