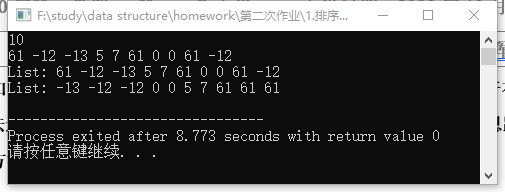
1、. 设元素值为整型的线性表L，分别采用顺序结构和链式结构存储，编写函数，用选择/冒泡排序算法实现线性表的表排序。

（1）选择排序：

算法思想：

双重循环，第i次选出一个最小的元素放在第i个位置

运行结果：



结果分析：

T(n) = O()

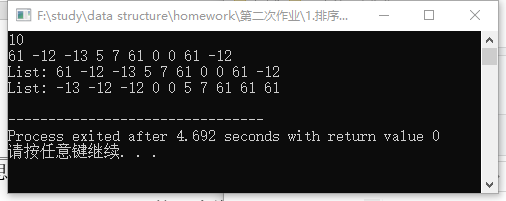
S(n) = O(1)

（2）冒泡排序：

算法思想：

双重循环，外层循环第i次将一个最大的元素放在倒数第i个位置，内层循环从前往后遍历数组，若前一个元素大于后一个元素则交换两个元素，循环结束后，从而将一个最大的元素放在倒数第i个位置。

运行结果：



结果分析：

T(n) = O()

S(n) = O(1)

附源程序。

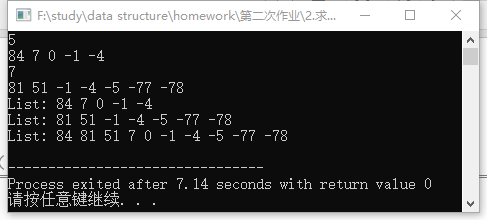
2、题目：设线性表A、B，元素值为整型，且递减有序，编写函数，实现下列功能：对采用顺序结构和链式结构2种存储结构，要求在A的空间上构成一个新线性表C，其元素为A和B元素的并集，且表C中的元素值递减有序（互不相同）。

（1）数组实现

算法思想：

定义int类型pos1、pos2，分别指向两链表表头，从头到尾同时遍历两链表，若pos2所指结点数据大于pos1所指，则插入，同时pos2自增，若等于，pos1、pos2同时自增，若小于p1自增。

运行结果：



结果分析：

T(n) = O(n\*m)

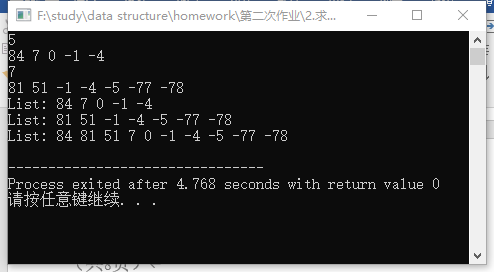
S(n) = O(1)

（2）链表实现

算法思想：

p1、p2分别指向两链表表头，从头到尾同时遍历两链表，若p2所指结点数据大于p1所指，则插入，同时后移p2，若等于，p1、p2同时后移，若小于p1后移。

运行结果：



结果分析：

T(n) = O(n+m)

S(n) = O(1)

附源程序。

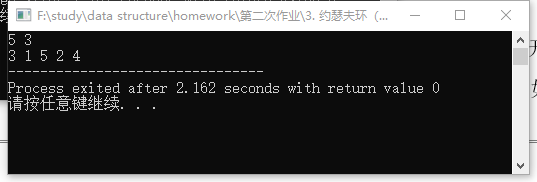
3、题目：输入正整数n、m（m<n），设有n个人坐成一圈，从第1个人开始循环报数，报到m的人出列，然后再从下一个人开始报数，报到m的人又出列，如此重复，直到所有的人都出列为止。要求用链式结构和顺序结构实现，按出列的先后顺序输出每个人的信息。

（1）数组实现

算法思想：

a[]用来存储第i个人的信息，out[]用来存储第i个人是否出列，m累计报数。一次循环检查一个元素，若报数为M，输出信息，使i出列，m置1，人数减1，若报数小于M，m加1。

运行结果：



结果分析：

T(n) = O(n\*m)

S(n) = O(1)

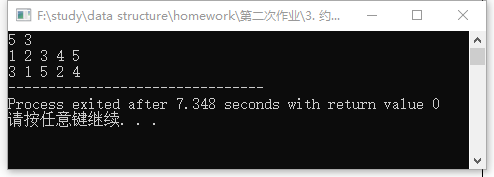
（2）链表实现

算法思想：

共两层循环。

第一层循环使得i遍历链表，第二层循环从i位置之后遍历链表，检查第i个元素重复元素的位置，若有则删去，若没有则返回空指针。

运行结果：



结果分析：

T(n) = O(n\*m)

S(n) = O(1)

附源程序。

4、题目：CSP 题目

题目描述：小明在他的果园里种了一些苹果树，这些苹果树排列成一个圆。为了保证苹果的品质，在种植过程中要进行疏果操作。为了更及时地完成疏果操作，小明会不时地检查每棵树的状态，根据需要进行疏果。检查时，如果发现可能有苹果从树上掉落，小明会重新统计树上的苹果个数（然后根据之前的记录就可以判断是否有苹果掉落了)。在全部操作结束后，请帮助小明统计相关的信息。

输入格式：

从标准输入读入数据。

第1行包含一个正整数N，表示苹果树的棵数。

第1+i行（1≤i≤N)，每行的格式为mi,ai1,ai2,... ,aimi,。其中，第一个正整数mi表示本行后面的整数个数。后续的mi个整数表示小明对第i棵苹果树的操作记录。若aij (1≤ j ≤mi）为正整数，则表示小明进行了重新统计该棵树上的苹果个数的操作，统计的苹果个数为aij;若为零或负整数，则表示一次疏果操作，去掉的苹果个数是|aij|。

输入保证一定是正确的，满足:

1. ai1>0，即对于每棵树的记录，第一个操作一定是统计苹果个数（初始状态，此时不用判断是否有苹果掉落）；
2. 每次疏果操作保证操作后树上的苹果个数仍为正。

输出格式：

输出到标准输出。

输出只有一行，包含三个整数T、D、E。其中，

1. T为全部疏果操作结束后所有苹果树上剩下的苹果总数（假设每棵苹果树在最后一次统计苹果个数操作后苹果不会因为疏果以外的原因减少);
2. D为发生苹果掉落的苹果树的棵数;
3. E为相邻连续三棵树发生苹果掉落情况的组数。

对于第三个统计量的解释：N棵苹果树A1,A2,...,AN排列成一个圆，那么A1与A2相邻，A2与A3相邻，......，AN-1与AN相邻，AN与A1相邻。如果Ai-1, Ai，Ai+1这三棵树都发生了苹果掉落的情况，则记为一组。形式化的，有

E=|{Ai/Drop(Pred(Ai)）∧Drop(Ai）∧Drop(Succ(Ai)）}|

其中，Drop(Ai）表示苹果树Ai是否发生苹果掉落的情况，Pred(Ai)表示Ai的前一棵树Ai-1（如果i>1）或者AN（如果i=1)，Succ(Ai)表示Ai的后一棵树Ai+1（如果i<N）或者A1（如果i=N）

算法思想：

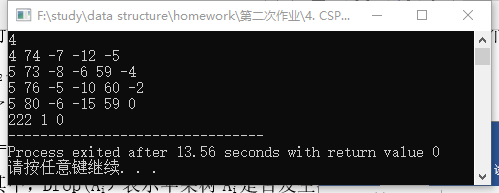
先开辟三个足够大的数组，Trees[]存放每棵树原始苹果个数，get[]每棵树疏果个数，loss[]每棵树掉落苹果个数。

计算T：Trees[]的总和减去get[]的总和，再减去loss[]的总和。

计算D：遍历loss[]，若loss[i]大于0，D加1。

计算E：定义两个flag，flag1为1表示上上棵树有掉落，flag2为1表示上棵树有掉落。遍历loss[]时，若loss[]大于0，令两个flag依次置1，若loss[]为0，两个flag均置0，若两个flag同时为1，E加1。

运行结果：



结果分析：

T(n) = O(n)

S(n) = O(n)

附源程序。

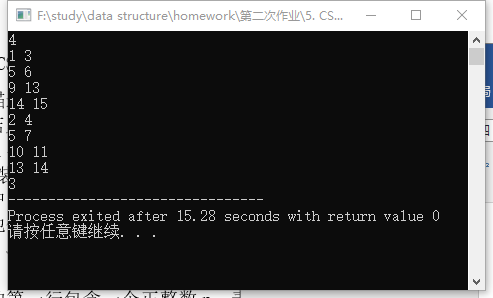
5、题目：CSP 题目

问题描述：小H和小W来到了一条街上，两人分开买菜，他们买菜的过程可以描述为，去店里买一些菜然后去旁边的一个广场把菜装上车，两人都要买n种菜，所以也都要装n次车。具体的，对于小H来说有n个不相交的时间段[a1,b1],[a2,b2]…[an,bn]在装车，对于小W来说有n个不相交的时间段[c1,d1],[c2,d2]…[cn,dn]在装车。其中，一个时间段[s, t]表示的是从时刻s到时刻t这段时间，时长为t-s。  
　　由于他们是好朋友，他们都在广场上装车的时候会聊天，他们想知道他们可以聊多长时间。  
输入格式  
　　输入的第一行包含一个正整数n，表示时间段的数量。  
　　接下来n行每行两个数ai，bi，描述小H的各个装车的时间段。  
　　接下来n行每行两个数ci，di，描述小W的各个装车的时间段。  
输出格式  
　　输出一行，一个正整数，表示两人可以聊多长时间。

算法思想：

先令time为0，并开辟一个足够大的数组H[],若i时间段小H在装车，H[i]置1，否则置0。当输入小W装车时间段时遍历H[],若为1,time累加，反之无操作。

运行结果：



结果分析：

T(n) = O(n)

S(n) = O(n)

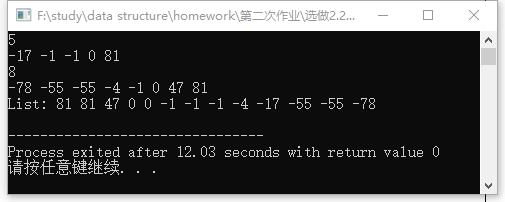
附源程序。

选做题2.24 **假设有两个按元素值递增有序排列的线性表A和B，均以单链表作存储结构，请编写算法将A表和B表归并成一个按元素值递减有序（即非递增有序，允许表中含有值相同的元素）排列的线性表C，并要求利用原表（即A表和B表）的结点空间构造C表。**

算法思想：

先分别逆转两个链表，之后同第2题的思路，唯一不同之处在于这个题允许元素重复，只要稍微修改“p1与p2所指元素相等”的情况即可。

运行结果：



结果分析：

T(n) = O(n+m)

S(n) = O(1)

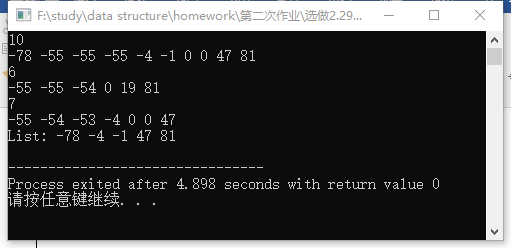
附源程序。

选做题2.29 **已知A，B和C为三个递增有序的线性表，现要求对A表作如下操作：删去那些既在B表中出现又在C表中出现的元素。试对顺序表编写实现上述操作的算法，并分析你的算法的时间复杂度（注意：题中没有特别指明同一表中的元素值各不相同）。**

算法思想：

先去除L2中L3没有的数据结点，结果保留在L2中，再去除L1中L2没有的数据结点，结果保留在L1中。

运行结果：



结果分析：

T(n) = O(len1\*len2+len2\*len3)

S(n) = O(1)

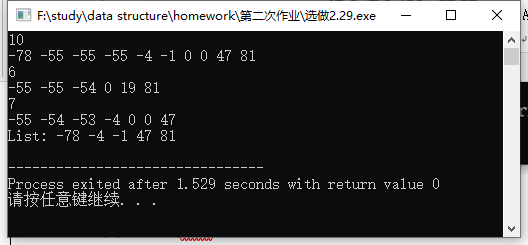
附源程序。

选做题**2.30 要求同2.29题。试对单链表编写算法，请释放A表中的无用结点空间。**

算法思想：

先去除L2中L3没有的数据结点，结果保留在L2中，再去除L1中L2没有的数据结点，结果保留在L1中。

运行结果：



结果分析：

T(n) = O(len1+len2+len3)

S(n) = O(1)

附源程序。