----

链表

教材第9章习题6~12，上机调试。

1.反转一个单链表。

示例:

输入: 1->2->3->4->5->NULL

输出: 5->4->3->2->1->NULL

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* reverseList(struct ListNode\* head){

}

2.给定一个排序链表，删除所有重复的元素，使得每个元素只出现一次。

示例 1:

输入: 1->1->2

输出: 1->2

示例 2:

输入: 1->1->2->3->3

输出: 1->2->3

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* deleteDuplicates(struct ListNode\* head){

}

3.给定一个排序链表，删除所有含有重复数字的节点，只保留原始链表中 没有重复出现 的数字。

示例 1:

输入: 1->2->3->3->4->4->5

输出: 1->2->5

示例 2:

输入: 1->1->1->2->3

输出: 2->3

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* deleteDuplicates(struct ListNode\* head){

}

4.给出两个非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照 逆序 的方式存储的，并且它们的每个节点只能存储 一位数字。如果我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。您可以假设除了数字0之外，这两个数都不会以0 开头。

示例：

输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)

输出：7 -> 0 -> 8

原因：342 + 465 = 807

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* addTwoNumbers(struct ListNode\* l1, struct ListNode\* l2){

}

5.给定一个单链表，把所有的奇数节点和偶数节点分别排在一起。请注意，这里的奇数节点和偶数节点指的是节点编号的奇偶性，而不是节点的值的奇偶性。

示例 1:

输入: 1->2->3->4->5->NULL

输出: 1->3->5->2->4->NULL

示例 2:

输入: 2->1->3->5->6->4->7->NULL

输出: 2->3->6->7->1->5->4->NULL

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* oddEvenList(struct ListNode\* head){

}

6.编写程序以x为基准分割链表，使得所有小于x的节点排在大于或等于x的节点之前。如果链表中包含x，x只需出现在小于x的元素之后（如下所示）。分割元素x只需处于“右半部分”即可，其不需要被置于左右两部分之间。

示例:

输入: head = 3->5->8->5->10->2->1, x = 5

输出: 3->1->2->10->5->5->8

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* partition(struct ListNode\* head, int x){

}

7.给你一个链表的头节点 head，请你编写代码，反复删去链表中由总和值为0的连续节点组成的序列，直到不存在这样的序列为止。删除完毕后，请你返回最终结果链表的头节点。

你可以返回任何满足题目要求的答案。

（注意，下面示例中的所有序列，都是对 ListNode 对象序列化的表示。）

示例 1：

输入：head = [1,2,-3,3,1]

输出：[3,1]

提示：答案 [1,2,1] 也是正确的。

示例 2：

输入：head = [1,2,3,-3,4]

输出：[1,2,4]

示例 3：

输入：head = [1,2,3,-3,-2]

输出：[1]

8.将两个升序链表合并为一个新的升序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

示例：

输入：1->2->4, 1->3->4

输出：1->1->2->3->4->4

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* mergeTwoLists(struct ListNode\* l1, struct ListNode\* l2){

}

9.请判断一个链表是否为回文链表。

示例 1:

输入: 1->2输出: false

示例 2:

输入: 1->2->2->1输出: true

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

bool isPalindrome(struct ListNode\* head){

}

案例3 电话订餐信息处理

仔细阅读理解教辅13.3节《案例3 电话订餐信息处理》的程序、原理、流程、算法等，然后进行以下程序改进、完善：

1、应该中午、晚上分别记最多接受订桌数。

2、应该按人名/手机号输入，不能按编号输入，

编号不要人工输入，应该自动编号；

应记录订桌时间。

3、查询功能不要按编号查询，

应该按人名、手机号码、时间等查询。

4、修改功能也不要按编号，要按人名、手机号等，

要先显示此条信息，再提示是否修改。

5、删除前也应显示此条信息，提示是否确定删除。

6、显示应该按日期/中午/晚上等分开列。

7、表格存到文件中，程序启动时先读文件，

程序结束时保存文件。

8、输入应该有错误重输、连续输入等功能。

不限于以上要求，还有哪些你认为有必要的修改？请详细说明并修改。

栈

1、假设以I和O分别表示入栈和出栈操作。栈的初态和终态均为空，入栈和出栈的操作序列可表示为仅由I和O组成的序列，可以操作的序列称为合法序列，否则称为非法序列。

1) 下面所示的序列中哪些是合法的？  
a. IOIIOIOO     b. IOOIOIIO     c. IIIOIOIO     d. IIIOOIOO  
2) 通过对1)的分析，写出一个算法，判定所给的操作序列是否合法。若合法，返回 true，否则返回false（假定被判定的操作序列已存入一维数组中）。

2、设单链表的表头指针为h，结点结构由data和next两个域构成，其中data域为字符型。试设计算法判断该链表的前n个字符是否中心对称。例如xyx、xyyx都是中心对称。

3、设有两个栈s1、s2都釆用顺序栈方式，并且共享一个存储区[0, …, maxsize-1]，为了尽量利用空间，减少溢出的可能，可釆用栈顶相向、迎面增长的存储方式。试设计s1、s2 有关入栈和出栈的操作算法。

4、这里有n列火车将要进站再出站，但是，每列火车只有1节，那就是车头。这n列火车按1到n的顺序从东方左转进站，这个车站是南北方向的，它虽然无限长，只可惜是一个死胡同，而且站台只有一条股道，火车只能倒着从西方出去，而且每列火车必须进站，先进后出。也就是说这个火车站其实就相当于一个栈，每次可以让右侧头火车进栈，或者让栈顶火车出站。

车站示意如图：

出站<—— <——进站

|车|

|站|

|\_\_|

现在请你按《字典序》输出前20种可能的出栈方案。

输入格式

输入一个整数n，代表火车数量。

输出格式

按照《字典序》输出前20种答案，每行一种，不要空格。

数据范围

1≤n≤201≤n≤20

输入样例：

3

输出样例：

123

132

213

231

321

5、请实现一种数据结构SetOfStacks，由多个栈组成，其中每个栈的大小为size，当前一个栈填满时，新建一个栈。该数据结构应支持与普通栈相同的push和pop操作。给定一个操作序列int[][2] ope，每个操作的第一个数代表操作类型，若为1，则为push操作，后一个数为应push的数字；若为2，则为pop操作，后一个数无意义。请返回一个int[][](C++为vector&ltvector&ltint>>)，为完成所有操作后的SetOfStacks，顺序应为从下到上，默认初始的SetOfStacks为空。保证数据合法。

6、用两个栈来实现一个队列，完成队列的Push和Pop操作。 队列中的元素为int类型。

7、你现在是棒球比赛记录员。给定一个字符串列表，每个字符串可以是以下四种类型之一：

1.整数（一轮的得分）：直接表示您在本轮中获得的积分数。

2. "+"（一轮的得分）：表示本轮获得的得分是前两轮有效 回合得分的总和。

3. "D"（一轮的得分）：表示本轮获得的得分是前一轮有效 回合得分的两倍。

4. "C"（一个操作，这不是一个回合的分数）：表示您获得的最后一个有效 回合的分数是无效的，应该被移除。

每一轮的操作都是永久性的，可能会对前一轮和后一轮产生影响。你需要返回你在所有回合中得分的总和。

示例：

输入: ["5","2","C","D","+"]

输出: 30

解释:

第1轮：你可以得到5分。总和是：5。

第2轮：你可以得到2分。总和是：7。

操作1：第2轮的数据无效。总和是：5。

第3轮：你可以得到10分（第2轮的数据已被删除）。总数是：15。

第4轮：你可以得到5 + 10 = 15分。总数是：30。

8、你将要实现一个功能强大的整数序列编辑器。在开始时，序列是空的。编辑器共有五种指令，如下：

1、“I x”，在光标处插入数值x。  
2、“D”，将光标前面的第一个元素删除，如果前面没有元素，则忽略此操作。  
3、“L”，将光标向左移动，跳过一个元素，如果左边没有元素，则忽略此操作。  
4、“R”，将光标向右移动，跳过一个元素，如果右边没有元素，则忽略次操作。  
5、“Q k”，假设此刻光标之前的序列为a1,a2,…,ana1,a2,…,an,输出max1≤i≤kSimax1≤i≤kSi，其中Si=a1+a2+…+aiSi=a1+a2+…+ai。

输入格式

第一行包含一个整数Q，表示指令的总数。接下来Q行，每行一个指令，具体指令格式如题目描述。

输出格式

每一个“Q k”指令，输出一个整数作为结果，每个结果占一行。

数据范围

1≤Q≤1061≤Q≤106,  
|x|≤103|x|≤103,  
1≤k≤n1≤k≤n

输入样例：

8

I 2

I -1

I 1

Q 3

L

D

R

Q 2

输出样例：

2

3

样例解释

下图包含了对样例的过程描述：

