2018/10/17　19：01

顺了一遍课本，发现对指针的操作还是有些陌生，对于一些删除操作，在结束时需要用 delete 运算符释放的是指针指向的存储空间，而不是指针本身.

2018/10/17　21：33

实现链表的删除操作时，发现了一个 bug：

.next()方法能够使 curr 指针指向尾节点，但若此时调用.getValue()方法时会抛出”No value”的异常，这是因为为了减少.insert()，.prev()方法的代码量，使 curr 指针指向了当前位置的前一个元素，所以在调用.next()方法时要同时判断当前位置是否是尾节点，若是，则取消这次的方法调用。

2018/10/18　20：16

在建立工程把所有文件包含在一起时，由于对#include 的使用不当，出现了文件重复包含的问题，比如：

#include"a.h"

#include"b.h"

看上去没什么问题。如果 a.h 和 b.h 都包含了一个头文件 x.h。那么 x.h 在此也同样被包含了两次，只不过它的形式不是那么明显而已。

多重包含在绝大多数情况下出现在大型程序中，它往往需要使用很多头文件，因此要发现重复包含并不容易。要解决这个问题，我们可以使用条件编译。如果所有的头文件都像下面这样编写:

#ifndef _HEADERNAME_H

#define _HEADERNAME_H

...//(头文件内容)

#endif

那么多重包含的危险就被消除了。当头文件第一次被包含时，它被正常处理，符号 HEADERNAME_H 被定义为 1。如果头文件被再次包含，通过条件编译，它的内容被忽略。符号 HEADERNAME_H 按照被包含头文件的文件名进行取名，以避免由于其他头文件使用相同的符号而引起的冲突。

但是，你必须记住预处理器仍将整个头文件读入，即使这个头文件所有内容将被忽略。由于这种处理将托慢编译速度，所以如果可能，应该避免出现多重包含。

2018/10/19　12：07

完成了附加题的解答，在解题过程中熟悉了链表的操作原理，在调用.remove()方法后，curr 指针会自动的向后移动一个位置，所以在下一个小朋友报数的时候，就无需再调用.next()方法了。