```
        ArrayList<Integer> lst = new ArrayList<Integer>( );

        for( int i = 0; i < N; i++ )
        {
            lst.add( i );
            lst.trimToSize( );
        }
    }
```

3.8 下列例程删除作为参数被传递的表的前半部分：

```
public static void removeFirstHalf( List<?> lst )
{
    int theSize = lst.size( ) / 2;

    for( int i = 0; i < theSize; i++ )
        lst.remove( 0 );
}
```

a. 为什么在进入 for 循环前存储 theSize?

b. 如果 lst 是一个 ArrayList，removeFirstHalf 的运行时间是多少?

c. 如果 lst 是一个 LinkedList，removeFirstHalf 的运行时间是多少?

d. 对于这两种类型的 List 使用迭代器都能使 removeFirstHalf 更快吗?

3.9 提供对 MyArrayList 类的 addAll 方法的实现。方法 addAll 将由 items 给定的特定集合的所有项添加到 MyArrayList 的末端。再提供上述实现的运行时间。你使用的方法声明与 Java Collections API 中的略有不同，其形式如下：

```
public void addAll( Iterable<? extends AnyType> items )
```

3.10 提供对 MyLinkedList 类的 removeAll 方法的实现。方法 removeAll 将由 items 给定的特定集合的所有项从 MyLinkedList 中删除。再提供上述实现的运行时间。你使用的方法声明与 Java Collections API 中的略有不同，其形式如下：

```
public void removeAll( Iterable<? extends AnyType> items )
```

3.11 假设单链表使用一个头节点实现，但无尾节点，并假设它只保留对该头节点的引用。编写一个类，包含

a. 返回链表大小的方法。

b. 打印链表的方法。

c. 测试值 x 是否含于链表的方法。

d. 如果值 x 尚未含于链表，添加值 x 到该链表的方法。

e. 如果值 x 含于链表，将 x 从该链表中删除的方法。

3.12 保持单链表以排序的顺序重复练习 3.11。

3.13 添加 ListIterator 对 MyArrayList 类的支持。java.util 中的 ListIterator 接口比 3.3.5 节所述含有更多的方法。注意，你要编写一个 listIterator 方法返回新构造的 ListIterator，并且还要注意现存的迭代器方法可以返回一个新构造的 ListIterator。这样，你将改变 ArrayListIterator，使得它实现 ListIterator 而不是 Iterator。对于 3.3.5 节中未列出的那些方法抛出 UnsupportedOperationException 异常。

3.14 如练习 3.13 所述，添加 ListIterator 对 MyLinkedList 类的支持。