线性表实现及应用

线性表提供了组织数据的一种模式，用它可以组织管理待办事项清单、礼品单、地址列表、甚至清单的清单等。线性表是一个基础性很强的数据结构，在很多的高级语言中，一般都会提供对这类数据组织方式的支持。

任务 1： 为指定的 List ADT 实现各种数据结构

在本次实验中，主要完成的任务是：

1、为指定的 List ADT（该 ADT 的具体要求请参见文件 List.java）实现三种数据结构：①使用顺序数组做为存储表示； ②使用单向链表做为存储表示； ③使用双向链表做为存储表示。不论使用哪种存储表示实现的数据结构，实验中需要处理的数据类型都是 Character 类型。

2、用给定的输入数据文件验证所实现的数据结构是否正确。

3、给出每种数据结构实现的每个方法在最坏情形下的时间复杂度表示。（使用表格列举结论）

为了方便进行测试验证，对 List 的各种操作指定了相应的命令符号，每个符号含义如下：

+	insert
-	remove
=	replace
#	gotoBeginning
*	gotoEnd
>	gotoNext
<	gotoPrev
~	clear

假如对初始化空间大小为 16 的空表按顺序依次执行每一行的命令列表，并且假设该线性表中插入的元素为 Character 类型，那么表格中第二列即为执行相对应的第一列中的命令列表后调用 showStructure 方法的运行结果（在阅读下面表格内容之前先仔细阅读 List.java 中对每个方法的详细说明，下一行的命令内容是接着上一行的命令内容执行）：

命令行内容	执行结果（调用 List 的 showStructure 方法）[1]
+a +b +c +d	a b c d {capacity = 16, length = 4, cursor = 3}
# > >	a b c d {capacity = 16, length = 4, cursor = 2}
*** < <**	a b c d {capacity = 16, length = 4, cursor = 1}
-	a c d {capacity = 16, length = 3, cursor = 1}
+e +f +f	a c e f f d {capacity = 16, length = 6, cursor = 4}
*** -**	a c e f f {capacity = 16, length = 5, cursor = 0}
-	c e f f {capacity = 16, length = 4, cursor = 0}
=g	g e f f {capacity = 16, length = 4, cursor = 0}
~	Empty list {capacity = 16, length = 0, cursor = -1}

实验完成之后，必须通过实验中提供的测试用例。借助测试用例的运行结果，用来检查所撰写

[1] 如果 List 实现的存储方式为链式结构，那么 capacity 的值即为真实的链表中的结点个数。实验文件中仅提供了基于数组实现的测试用例的运行结果。

的代码功能是否正确。测试用例中的每一行的内容都类似于上表中的每一行“命令行内容”列中所指示的内容。要求每执行一行，就调用 List 接口中的 showStructure 行为，用以验证该命令行的执行是否正确。每行“命令行内容”都不是独立的，是针对同一个 List 类型的对象实例运行的结果。实验包里包括了个文件，一个是“list_testcase.txt”，其内包含了测试用例；另一个是“list_result.txt”，其内包含了对应测试用例的运行结果。

任务 2： 为指定的 DQueue ADT 实现两种数据结构

在本次实验中，完成双端队列抽象数据类型的实现。双端队列是普通队列的一个扩展，其允许在队头和队尾执行入队和出队操作，因此双端队列兼具了栈和队列的操作能力。

1、为指定的 DQueue ADT（该 ADT 的具体要求请参见文件 DQueue.java）实现两种数据结构：①使用顺序数组做为存储表示； ②使用双向链表做为存储表示。不论使用哪种存储表示实现的数据结构，实验中需要处理的数据类型都是 Integer 类型，且对 ADT 中的每个行为要求在最坏情形下的时间复杂度都必须满足 O(1)。

2、借助合适的数据结构解决如下问题。

问题描述：给定一个包含 n（n>0）个数据的序列，和一个 K（0<K<=n）值，要求计算序列中每 K 个连续元素构成的子序列的最大值。

资源要求：最坏情形下的时间复杂度为 O(n)。

举例：当数据序列为：8、5、10、7、9、4、15、12、90、13 和 K=4，那么每 K 个连续的子序列的最大值分别为：10、10、10、15、15、90 和 90。

随机生成 10 组数据完成测试。

任务 3： 栈

众所周知，栈虽然是一个操作受限的线性表，但是其用途却很广泛，栈的数据结构实现非常简单，因此我们只从应用层面熟悉栈。请完成下面三个子任务。

①递归是一种解决很多复杂问题最简单的思想方法，而任何编程语言对递归程序的支持都是通过栈实现的。请利用课堂上讲解的“Hanoi 塔”问题的非递归转化方法完成对递归快速排序的非递归转化。

②算术混合运算表达式的计算。表达式不仅能处理整数，还需要处理小数。表达式中涉及的运算符包括+、-、*、/、^(指数)。表达式可以包含括号（只包含圆括号）嵌套，因此要处理括号匹配失败的情形。

③寻找连续的股票最大走势区间。在股票市场中，需要识别股价走势中的连续时间段，这些时间段内股价表现出特定的行为模式，如连续上涨、连续下跌或是横盘整理等。这种分析对于技术分析尤为重要，它帮助交易者和投资者理解市场的动向，判断趋势的持续性或反转的可能性。给定一个记录股票价格的数组 prices，需要生成对应的 s 数组，s[i]中记录的值为 i-j+1，其中 i 和 j 必须要满足 prices[j..i-1]中的每一个值都符合 prices[j]<=prices[i]。s 数组中的最大值即为连续上涨的最大区间，如图 1 所示。

设计并实现一个算法完成对给定的 prices 数组上求最大连续上涨时间区间值。如果仅使用朴实的设计思想，该算法的时间复杂和空间复杂度是多少？如果使用栈设计的算法，其时间复杂度和空间复杂度是多少？

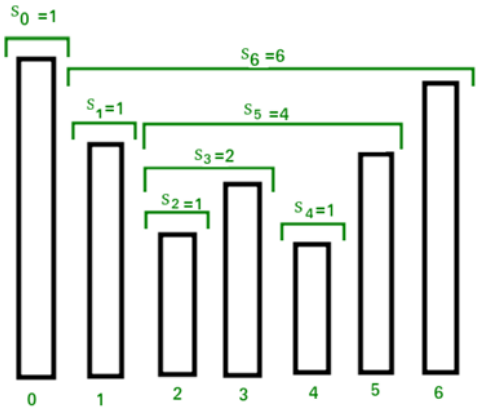

图 1：以 prices=100,80,60,70,60,75,85 为例的 s 的值

任务 4：基数排序

使用自定义的队列数据结构实现对某一个数据序列的排序(采用基数排序)，其中对待排序数据有如下的要求：

①当数据序列是整数类型的数据的时候，数据序列中每个数据的位数不要求等宽，比如： 1、21、12、322、44、123、2312、765、56

②当数据序列是字符串类型的数据的时候，数据序列中每个字符串都是等宽的，比如："abc","bde","fad","abd","bef","fdd","abe"

注：radixsort1.txt 和 radixsort2.txt 是为上面两个数据序列提供的测试数据。