线性表实现及应用

线性表提供了组织数据的一种方法，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。这些清单为人们有条理的安排生活提供了一定的保障。所以说，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 ： 实现线性表的基本操作

在本次实验中， 主要完成的任务是:

1、基于线性表的两种存储结构（即顺序存储和链式存储）编程实现以下功能:

InitList(&L):初始化表。构造一个空的线性表。

DestroyList(&L):销毁表。销毁线性表L并释放线性表所占存储空间。

Length(L):求表长。返回线性表L的长度，即线性表L中元素的个数。

Empty(L):判空。若L为空表则返回True,否则返回Flase。

PrintList(L):输出表。将线性表的元素依次输出。

LocateList(L, e):按值查找操作，查找元素e在表L中的位置。

GetList(L, i):按位查找操作，查找第i个位置的元素的值。

ListInsert(&L, i, e):在表L的第i个位置插入元素e。

ListDelete(&L, i, &e):删除表L的第i个位置的元素，并用e返回删除元素的值。

2、要求对于每一种操作进行算法设计描述、实现描述及测试说明（须有程序运行截图）。

3、比较并说明上述操作在顺序存储和链式存储下的异同。

任务 2：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。 请自行查阅资料理解 “Hanoi 塔”问题，通过递归编程的方式解决该问题，并尝试使用非递归的编程方法解决该问题。

②请在①的基础上进一步学习掌握递归问题的非递归转化方法，并实现对递归快速排序的非递归转化。

③当我们在使用很多软件时都有类似“undo”功能，比如 Web 浏览器的回退功能、文本编辑器的撤销编辑功能。这些功能都可以使用 Stack 简单实现， 但是在现实中浏览器的回退功能也好，编辑器的撤销功能也好， 都有一定的数量限制。因此我们需要的不是一个普通的 Stack 数据结构，而是一个空间有限制的 Stack，虽然空间有限，但这样的 Stack 在入栈时从不会溢出，因为它会采用将最久远的记录丢掉的方式让新元素入栈， 也就是说总是按照规定的数量要求保持最近的历史操作。比如栈的空间是 5，当 a\b\c\d\e 入栈之后，如果继续让元素 f 入栈，那么栈中的元素将是 b\c\d\e\f。请设计一个满足上面要求的 LeakyStack 数据结构， 要求该数据结构的每一个操作的时间复杂度在最坏情形下都必须满足 O(1)。

任务 3：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。