# LintCode详解

## 链表

### 删除链表中的元素

题目大意：删除链表中等于给定值val的所有节点。

解题思路：声明dummy作为单链表的头节点，只需要一个指针head指向待删除节点的前节点，从dummy开始，判断指针后节点的数值是否等于val，循环终止的条件为head.next != null。

**注意：**

1. 声明dummy节点：以便对“1🡪null”的处理。
2. 一个指针：删除节点必须记录删除前的位置，可以不用记录待删除节点的位置；若指针记录删除节点的位置，则尾节点无法删除。
3. 循环终止的条件。

代码：170927\_452\_removeElements.java

### 删除链表中的元素

题目大意：2🡪1🡪5🡪null和5🡪9🡪2🡪null，输出和7🡪0🡪8🡪null。

解题思路：循环中用sum记录上一次进位的情况，如果list1不为空，加上值，**接着**，如果list2不为空，加上值，一次循环完，修改sum的值为其十位数。

**注意：**

1. 情况分析，易忽略进位的情况“空，空，进位”。
2. 顺序if，见Java记录一。

### 合并两个排序链表

题目大意：将两个排序链表合并为一个新的排序链表。1🡪3🡪8🡪11🡪15🡪null和2🡪null合并为1🡪2🡪3🡪8🡪11🡪15🡪null。

解题思路：“合并”想到“**递归**”。

*Ps:对于连续的相同的操作，除了迭代循环，可以考虑是否有递归的方式。*

**注意：**递归时指针移动的处理。

## 哈希表

哈希（Hash）是一种数据编码方式，将大尺寸的数据（如一句话，一张图片，一段音乐、一个视频等）浓缩到一个数字中，从而方便地实现数据匹配·查找的功能。

哈希算法并不是一个特定的算法而是一类算法的统称。哈希算法也叫散列算法，一般来说满足这样的关系：f(data)=key，输入任意长度的data数据，经过哈希算法处理后输出一个定长的数据key。同时这个过程是**不可逆**的，无法由key逆推出data。

Eg: 比如这里有一万首歌，要求按照某种方式保存好。到时候给你一首新的歌（命名为X），要求你确认新的这首歌是否在那一万首歌之内。无疑，将一万首歌一个一个比对非常慢。但如果存在一种方式，能将一万首歌的每一首的数据浓缩到一个数字（称为哈希码）中，于是得到一万个数字，那么用同样的算法计算新的歌X的编码，看看歌X的编码是否在之前那一万个数字中，就能知道歌X是否在那一万首歌中。将一首歌的5M字节数据浓缩到一个数字中的算法就是哈希算法。那一万首歌按照各自的编码数字从小到大排序后得到的一个表就是哈希表。显然，由于信息量的丢失，有可能多首歌的哈希码是同一个。好的哈希算法会尽量减少这种冲突，让不同的歌有不同的哈希码。最差的哈希算法自然就是所有的歌用那个算法算出来的都是同一个哈希码。

# Java知识点补充（LintCode）

## if、if…else if、if…if

if…else if：多重if，在条件1不满足的情况下，才会进行条件2的判断。

if…if：多个判断，顺序执行。

## 对象声明

p.next = new ListNode(sum % 10);

## int转String，再每位平方

|  |  |
| --- | --- |
| 1  2  3  4  5 | int n **=** 324**;**  String cn **=** n **+** ""**;** //int转String的常用方法  **for** **(**int i**=**0**;** i **<** cn**.**length**();** i**++){**  m **+=** **(**Integer**.**parseInt**(**cn**.**substring**(**i**,**i**+**1**)))\*(**Integer**.**parseInt**(**cn**.**substring**(**i**,**i**+**1**)));** //Java中^2表示亦或，而不是平方  **}** |

次方：

import java.lang.Math;

Math.pow(double a,double b) 返回第一个参数的第二个参数次幂的值。

## String和char[]的长度

String .length**()**;

char[] .length;