数字与字符串相互转换，简单题目中的隐藏陷阱

在算法面试中，很常见的考察题目是将数字转换为对应的字符串，或将字符串转换为数字，也就是让你实现 c++的库函数 atoi，或 itoa.这类题目要实现不难，但很多情况下，就算你给出了算法实现，但很可能也达不到面试官的要求，那是因为此类题目表明看起来不难，但实现时会有一些陷阱或坑，一旦掉入坑内，要想通过面试官的考核就有点问题了，下面我们分析两道该类算法题，看看我们应该如何避免这类问题所蕴含的陷阱。

写一个函数，功能是实现数字不同进制间的转换，函数的第一个输入参数是代表数字的字符串 s，第二个是一个整形 b1，表示 s 所代表的数字的进制，第三个参数也是一个整形 b2，表示要转换的进制,其中 1<b2,b1<=16。假设这个函数名为 numberConvert，那么 numberConvert(“100”，10，16)返回的是 100 的 16 进制表示：64.

处理这道题时，一定要注意，s 所代表的数字不一定是十进制的，有可能是十二进制或八进制等。不同进制间的转换，最好的处理方法是先把进制为 b1 的数转换成十进制，然后再把对应的十进制数转换成进制为 b2 的数。

假设字符串”a(k-1) a(k-2)… a(0)” 代表的是 b 进制的数，那么该字符串对应的十进制数可以这么算：$\sum_{i=0}^{k-1} a_i b^i$. 根据上头公式，我们可以实现一个将给定进制数转换为十进制数的函数:

```
int strToInt(String s, int b) {
    int val = 0, base = 1;
    for (int i = s.length() - 1; i >= 0; i--) {
        char c = s.charAt(i);   //a(i)
        int v = 0;
        if ('0' <= c && c <= '9') {
            v = c - '0';
        } else if (c >= 'A' && c <= 'E') {
            v = 10 + c - 'A';
        }

        if (i < s.length() - 1) {
            base *= b; //b^i
        }
        val += v * base;
    }
```

```
    return val;
}
```

大家测试一下上面的代码:

strToInt(“20”, 16) 结果为 32.

StrToInt(“1B”, 13) 结果为24,(24的13进制表示为1B).

若干次测试，输入相应的数字字符串和对应的进制数，函数都返回正确的 10 进制数。那么大家想想，上面的实现还有什么问题吗。

如果你找不到上面代码的问题，那表明你可能就要掉到坑里去了。一些看似实现不难的算法，它往往隐含一些非常容易出错的边界条件，以上面代码为例，如果 s 代表的是一个负数”-20”, 如果 s 是非法字符串例如”123gh”, 如果 s 的形式是:”+20”, 如果 b 的值小于 1 或大于 16，那么在这些边界条件下，我们上面的代码就要出问题。我曾经拿此类题目去面试候选人，百分之九十五的候选人能写出上面的代码但都无法对边界条件进行周全的考虑。所以边界判断是算法面试中要非常注意的事项，他们也是算法面试中的陷阱所在，当你发现算法实现比较简单时，那就是要提高警惕，注意处理边界条件，以防掉入面试官的考察陷阱，最典型的是

让你实现 strcpy(char*src, char* dst)，这种考察的目的根本不在于你是否会不会实现字符串拷贝的算法原理，重点就在于考察你对边界条件的判断，你想通过考察，你在代码中药判断两个指针是否为空，要判断两个指针是否相等，相等就无需做任何拷贝。上面的边界处理留个读者进行作业。

接下来我们再看看，十进制数如何转换为其他进制的数，假定我们要转换的进制用 b 来表示。根据前面讲到的公式，如果要转换的十进制数我们用 v 表示，那么就有:

v = a(k-1)*bˆ(k-1) + a(k-2)*bˆ(k-2)+… a(1)*b + a(0)*bˆ(0).

目的是想要得到 a(k-1)..a(0)，根据上面式子，不难发现，a(0) = v%b.

v1 = v /b = a(k-1)*bˆ(k-1) + a(k-2)*bˆ(k-2)+… a(1)

a(1) = v1 % b;

依次类推可得 a(k-1)到 a(k0).

于是我们便有一下代码：

```
String intToStr(in val, int base) {
    String s = “”;
    while (val > 0) {
        int d = val % base;
```

```
            char c = '0';
            if (d >= 0 && d <= 9) {
                c = (char)('0' +d);
            }
            else if (d >= 10) {
                c = (char)('A' + d - 10);
            }

            String t = "";
            t += c;
            s = t + s;
            val /= base;
        }
        return s;
    }
```

上面的代码，虽然逻辑都实现了，但有了前面的经验，大家知道代码还存在哪些问题了吧。边界条件的处理，体现了工程师心思慎密的程度，在面试中，往往用来将合适的候选人筛选出来。

我们再看看第二题：

类似于 excel 的应用，它们都喜欢用字母来对每一列进行编

码，例如第一列编码为”A”，第二列为”B” ….“Z”，“AA”，“AB”…..”ZZ”，“AAA”,”AAB”…

其中的 A 对应于 1，要求你实现一个函数，

int ssDecodeColID(String s)

输入是上面的编码字符串，返回该字符串对应的整形数值。

有了前道题的解析，这道题大家处理起来应该不难。把他们先转换成十进制看看，”Z”等价于 26，“AA”等价于 27，由此看，上面的编码，实际上是以 26 为进制的编码系统，并且该系统不存在数字 0.我们先把字母转换成他们对应的数字，例如

“ABC” -> 123

由于是 26 进制，因此 123 转换成十进制就有：

3*26^0 + 2*26^1 + 1 * 26^2 = 731.

于是实现代码如下：

```
int  ssDecodeColID(String s) {
    int pos = s.length() -1;
    int base = 1;
    int val = 0;

    while (pos >= 0) {
```

```
            int d = s.charAt(pos) - 'A' + 1;
            if (pos < s.length() - 1) {
                base * = 26;
            }
            val += d * base;
            pos--;
        }

        return val;
    }
```

上面的代码虽然算法逻辑都实现了，但它存在什么问题，想必读者您也心中有数了吧，请大家尝试着把上面代码存在的缺陷给弥补了。