题意：

已知一个数和字符串，求最小根r使得字符串在该根下的值刚好等于已知数。

分析：

本题求解思路不难想出，问题在于繁琐的细节处理。

大体思路就是用二分查找寻满足条件的根，匹配条件依据就是串数转换(参见函数my\_atoi)。

另外，二分查找的上界和下界决定了算法实际运行的效率。

不难想到：根的下界最小为2(参见Debug报告)，上界最大为欲匹配数tar + 1。

实际上，上界还可以更精确一些，在处理长字符串的时候下述特性十分好用：

假设：字符串s长度为l，以r为根，其值为t，函数m(c)表示将字符c映射为数字，应满足以下关系：

t = m(s[0]) \* r^(len - 1) + m(s[1]) \* r^(len - 2) + … +m(s[len - 1]) \* r^0；

=> m(s[0]) \* r^(len - 1) <= t，其中m(s[0])= 1

=> r^(len - 1) <= t

=> r <= t^(1 / (len - 1))

又知r < t + 1,

故r的上界为MIN(t^(1 / (len - 1), t + 1)，len为整数，当len > 2时，显然r的上界为t^(1 / (len - 1))(参见L46)。

对于r的下界至少为字符串s中值最大那位+1，如串“a1a1”,其r值至少为11，否则为不合法串(关于下界的处理是我在写\*wrong.c的时候有2个case过不去，翻看曾经的\*old.c发现倪端 参见L44 Debug报告)。

注意：

要用64位整型。原因是对于长度为10的串，根大小超过8，32位整型就hold不住了。会有很多case卡在数据类型上。

对于uint64的幂的计算pow的精度不够，同样有几个case无法处理。

无法使用pow，如何紧确r的上界呢：

我们假定t < 2^64 (uint64能hold住 值域为[0,2^64 - 1])

于是r^(len - 1) <= t < 2^64

=> r < 2^(64 / (len - 1))

2的幂用左移的方式表示，对于长整形不要忘记后面加LL。

另外，不要忘记len <= 2的情形：

len = 1除0非法；len = 2溢出非法(参见L47)。

上述每个细节处理不好都会被case卡...

剩下的细节就是在二分的时候用tmp寄存my\_atoi的计算结果，否则，在一次二分处理时会调两次函数(参见L49-L60 对比\*wrong.c这个细节我没处理)。

同样的细节在L47也没处理，只是因为我想少写几行代码。