43、N 进制减法，考点 or 实现——字符串，数组，集合操作

题目描述

主管期望你实现一个基于字符串的N进制的减法。

需要对输入的两个字符串按照给定的N进制进行减法操作，输出正负符号和表示结果的字符串。

输入描述

输入有三个参数：

1. 第一个参数是整数形式的进制N值，N值范围为大于等于2、小于等于35。
2. 第二个参数为被减数字符串；
3. 第三个参数为减数字符串。

有效的字符包括0-9以及小写字母a-z，字符串有效字符个数最大为100个字符，另外还有结尾的\0。

限制：

输入的被减数和减数，除了单独的0以外，不能是以0开头的字符串。

如果输入有异常或计算过程中有异常，此时应当输出-1表示错误。

输出描述

输出有2个。

其一为减法计算的结果，-1表示出错，0表示结果为整数，1表示结果为负数。

其二为表示结果的字符串。

用例

输入	2 11 1
输出	0 10
说明	按二进制计算 11 -1 ，计算正常，0表示符号为正数，结果为10

输入	8 07 1
输出	-1
说明	按8进制，检查到减数不符合非0前导的要求，返回结果为-1，没有其他结果内容。

题目解析

我是这么思考的，先不考虑异常情况，做N进制减法运算，最简单的就是将N进制数转为十进制，然后进行十进制减法，得到的差值，再转为N进制数。

而刚好JS语言中，可以利用Number.parseInt(str, n)，将n进制字符串数值转为十进制数，然后可以利用Number.prototype.toString(n)，再将十进制数转为N进制数值字符串。

有了以上思路后，本题几乎就解决了。

但是，本题中给了几种异常情况，并且题目要求异常返回-1。

因此我们需要注意异常情况的处理。

1、N值范围为大于等于2、小于等于35

2、减数、被减数的有效的字符包括0-9以及小写字母a-z

3、减数、被减数字符串有效字符个数最大为100个字符

4、输入的被减数和减数，除了单独的0以外，不能是以0开头的字符串。

另外还有一个隐藏的异常就是，一个N进制数，它的每一位不可能大于等于N，这也是我们需要考虑的异常情况

JavaScript算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const [n, shou, gong] = line.split(" ");
  console.log(getResult(n - 0, shou, gong));
});

/**
 *
 * @param {*} n 表示shou和gong是n进制数
 * @param {*} shou 被减数
 * @param {*} gong 减数
 */
function getResult(n, shou, gong) {
  if (n < 2 || n > 35) return -1;
  if (!valid(shou, n) || !valid(gong, n)) return -1;

  shou = parseInt(shou, n);
  gong = parseInt(gong, n);

  const diff = Number(Math.abs(shou - gong)).toString(n);

```

```
  return `${shou >= gong ? 0 : 1} ${diff}`;
}

function valid(str, n) {
  // 含前导的0只有0值本身合法
  if (str.startsWith("0")) {
    return str === "0";
  }

  // 被减数，减数只能包含字符0-9，a-z
  const regExp = /[^a-z0-9]/;
  if (regExp.test(str)) return false;

  // 被减数，减数长度最多100
  if (str.length > 100) return false;

  // 被减数，减数的每位不能超过n
  for (let c of str) {
    if (/^[0-9]$/.test(c)) {
      if (c - 0 >= n) return false; // 比如2进制数的每一位不能超过2
    } else {
      if (String(c).charCodeAt() - 87 >= n) return false; // ，16进制数每一位不能超过f
    }
  }

  return true;
}
```

Java算法源码

```
import java.util.Scanner;
import java.util.regex.Pattern;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();
    String b_sub = sc.next();
    String sub = sc.next();
    System.out.println(getResult(n, b_sub, sub));
  }

  /**
   * @param n 表示被减数和减数是n进制数
   * @param b_sub 被减数
   * @param sub 减数
   * @return 输出正负符号和表示结果的字符串
   */
  public static String getResult(int n, String b_sub, String sub) {
    if (n < 2 || n > 35) return "-1";

    if (!isValid(b_sub, n) || !isValid(sub, n)) return "-1";

```

```
    long b_sub_val = Long.parseLong(b_sub, n);
    long sub_val = Long.parseLong(sub, n);

    String diff = Long.toString(Math.abs(b_sub_val - sub_val), n);
    String sign = b_sub_val >= sub_val ? "0" : "1";

    return sign + " " + diff;
  }

  public static boolean isValid(String str, int n) {
    // 含前导的0只有0值本身合法
    if (str.startsWith("0")) return "0".equals(str);

    // 被减数，减数只能包含字符0-9，a-z
    Pattern reg = Pattern.compile("[^a-z0-9]");
    if (reg.matcher(str).find()) return false;

    // 被减数，减数长度最多100
    if (str.length() > 100) return false;

    // 被减数，减数的每位不能超过n
    for (int i = 0; i < str.length(); i++) {
      char c = str.charAt(i);
      if (c >= '0' && c <= '9' && Integer.parseInt(c + "") >= n) return false; // 比如2进制数的每一位不能超过2
      if (c >= 'a' && c <= 'z' && c - 87 >= n) return false; // 比如16进制数每一位不能超过f
    }

    return true;
  }
}
```

Python算法源码

```
import re

# 输入获取
n, b_sub, sub = input().split()

covertStr = "0123456789abcdefghigklmnopqrstuvwxyz"


# 将十进制数num转为base进制数
def toBase(num, base):
    if num < base:
        return covertStr[num]
    else:
        return toBase(num // base, base) + covertStr[num % base]


def isValid(s, n):
    #  含前导的0只有0值本身合法
    if s.startswith("0"):
        return s == "0"

    # 被减数，减数只能包含字符0-9，a-z
    if re.search("[^0-9a-z]", s) is not None:
        return False

    # 被减数，减数长度最多100
```

```
    if len(s) > 100:
        return False

    # 被减数，减数的每位不能超过n
    for c in s:
        if c.isnumeric() and int(c) >= n:  # 比如2进制数的每一位不能超过2
            return False
        elif c.isalpha() and ord(c) - 87 >= n:  # 比如16进制数每一位不能超过f
            return False

    return True


# 算法入口
def getResult(n, b_sub, sub):
    n = int(n)
    if n < 2 or n > 35:
        return "-1"

    if not isValid(b_sub, n) or not isValid(sub, n):
        return "-1"

    b_sub_val = int(b_sub, n)
    sub_val = int(sub, n)

    diff = toBase(abs(b_sub_val - sub_val), n)
    sign = "0" if b_sub_val >= sub_val else "1"
```

```
    return sign + " " + str(diff)


# 算法调用
print(getResult(n, b_sub, sub))
```