

```

        return 0;
    }
    char[] charArr = str.toCharArray();
    int res = 0;
    int num = 0;
    boolean posi = true;
    int cur = 0;
    for (int i = 0; i < charArr.length; i++) {
        cur = charArr[i] - '0';
        if (cur < 0 || cur > 9) {
            res += num;
            num = 0;
            if (charArr[i] == '-') {
                if (i - 1 > -1 && charArr[i - 1] == '-') {
                    posi = !posi;
                } else {
                    posi = false;
                }
            } else {
                posi = true;
            }
        } else {
            num = num * 10 + (posi ? cur : -cur);
        }
    }
    res += num;
    return res;
}

```

去掉字符串中连续出现 k 个 0 的子串

【题目】

给定一个字符串 `str` 和一个整数 k ，如果 `str` 中正好有连续的 k 个 '0' 字符出现时，把 k 个连续的 '0' 字符去除，返回处理后的字符串。

【举例】

`str="A00B"`， $k=2$ ，返回 `"A00B"`。

`str="A0000B000"`， $k=3$ ，返回 `"A0000B"`。

【难度】

士 ★☆☆☆

【解答】

解决本题能做到时间复杂度为 $O(N)$ 、额外空间复杂度为 $O(1)$ 的方法有很多。本书仅提供一种供读者参考。解法的关键是如何在从左到右遍历 `str` 时，将正好有连续的 k 个 '0' 的字符串都找到，然后把字符 '0' 去掉。具体过程如下：

1. 生成两个变量。整型变量 `count`，表示目前连续个 '0' 的数量；整型变量 `start`，表示连续个 '0' 出现的开始位置。初始时，`count=0`，`start=-1`。

2. 从左到右遍历 `str`，假设遍历到 i 位置的字符为 `cha`，根据具体的 `cha` 有不同的处理。

3. 如果 `cha` 是字符 '0'，令 `start = start == -1 ? i : start`，表示如果 `start` 等于 -1，说明之前没处在发现连续的 '0' 的阶段，那么令 `start=i`，表示连续的 '0' 从 i 位置开始，如果 `start` 不等于 -1，说明之前就已经处在发现连续的 '0' 的阶段，所以 `start` 不变。令 `count++`。

4. 如果 `cha` 不是字符 '0'，是去掉连续 '0' 的时刻。首先看此时 `count` 是否等于 k ，如果等于，说明之前发现的连续 k 个 '0' 可以从 `start` 位置开始去掉，如果不等于，说明之前发现的连续的 '0' 数量不是 k 个，则不能去掉。最后令 `count=0`，`start=-1`。

5. 既然把去掉连续 '0' 的时机放在了 `cha` 不是字符 '0' 的时候，那么如果 `str` 是以字符 '0' 结尾的，可能会出现最后一组正好有连续的 k 个 '0' 字符出现而没有去掉的情况。所以遍历完成后，再检查一下 `count` 是否等于 k ，如果等于，就去掉最后一组连续的 k 个 '0'。

具体过程请参看如下代码中的 `removeKZeros` 方法。

```
public String removeKZeros(String str, int k) {
    if (str == null || k < 1) {
        return str;
    }
    char[] chas = str.toCharArray();
    int count = 0, start = -1;
    for (int i = 0; i != chas.length; i++) {
        if (chas[i] == '0') {
            count++;
            start = start == -1 ? i : start;
        } else {
            if (count == k) {
                while (count-- != 0)
                    chas[start++] = 0;
            }
            count = 0;
            start = -1;
        }
    }
    if (count == k) {
        while (count-- != 0)
            chas[start++] = 0;
    }
}
```

```

    }
    return String.valueOf(chas);
}

```

判断两个字符串是否互为旋转词

【题目】

如果一个字符串 `str`，把字符串 `str` 前面任意的部分挪到后面形成的字符串叫作 `str` 的旋转词。比如 `str="12345"`，`str` 的旋转词有"12345"、"23451"、"34512"、"45123"和"51234"。给定两个字符串 `a` 和 `b`，请判断 `a` 和 `b` 是否互为旋转词。

【举例】

`a="cdab"`，`b="abcd"`，返回 `true`。

`a="1ab2"`，`b="ab12"`，返回 `false`。

`a="2ab1"`，`b="ab12"`，返回 `true`。

【要求】

如果 `a` 和 `b` 长度不一样，那么 `a` 和 `b` 必然不互为旋转词，可以直接返回 `false`。当 `a` 和 `b` 长度一样，都为 N 时，要求解法的时间复杂度为 $O(N)$ 。

【难度】

士 ★☆☆☆

【解答】

本题的解法非常简单，如果 `a` 和 `b` 的长度不一样，字符串 `a` 和 `b` 不可能互为旋转词。如果 `a` 和 `b` 长度一样，先生成一个大字符串 `b2`，`b2` 是两个字符串 `b` 拼在一起的结果，即 `String b2 = b + b`。然后看 `b2` 中是否包含字符串 `a`，如果包含，说明字符串 `a` 和 `b` 互为旋转词，否则说明两个字符串不互为旋转词。这是为什么呢？举例说明，假设 `a="cdab"`，`b="abcd"`。`b2="abcdabcd"`，`b2[0..3]="abcd"`是 `b` 的旋转词，`b2[1..4]="bcda"`是 `b` 的旋转词……`b2[i..i+3]`都是 `b` 的旋转词，`b2[4..7]="abcd"`是 `b` 的旋转词。由此可见，如果一个字符串 `b` 长度为 N 。在通过 `b` 生成的 `b2` 中，任意长度为 N 的子串都是 `b` 的旋转词，并且 `b2` 中包含字符串 `b` 的