题目描述：

给你一个字符串 s，字符串 s 首尾相连成一个环形 ，请你在环中找出 'o' 字符出现了偶数次最长子字符串的长度。

输入描述：

输入是一串小写字母组成的字符串

输出描述：

输出是一个整数

补充说明：

1 <= s.length <= 5 x 10^5

s 只包含小写英文字母。

示例 1

输入：

```
alolobo
```

输出：

```
6
```

说明：

最长子字符串之一是 "alolob"，它包含'o' 2 个。

示例 2

输入：

```
looxdolx
```

输出：

```
7
```

说明：

最长子字符串是 "oxdolxl"，由于是首尾连接在一起的，所以最后一个 'x' 和开头的 'l'是连接在一起的，此字符串包含 2 个'o' 。

示例 3

输入：

```
bcbcbc
```

输出：

```
6
```

说明：

这个示例中，字符串 "bcbcbc" 本身就是最长的，因为 'o' 都出现了 0 次。

```
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextLine()) { // 注意 while 处理多个 case
            String data = in.nextLine();
            System.out.println(parseChars2(data));
            // System.out.println(parseChars(data));
        }
    }
    public static int parseChars2(String data) {
        // 存在多少个 o
        int count = data.length() - data.replaceAll("o", "").length();
        if (count % 2 == 0) {
            // 总体偶数个
            return data.length();
        }
        return data.length() - 1;
    }

    public static int parseChars(String data) {
        String maxSub = "";

        for (int i = 0; i < data.length(); i ++ ) {
```

```
                for (int j = i + 1; j <= data.length() + i; j ++) {
                    String sub = "";
                    if (j > data.length()) {
                        sub = data.substring(i) + data.substring(0, j - data.length());
                    } else {
                        sub = data.substring(i, j);
                    }
                    String tmp = sub.replaceAll("o", "");
                    if (sub.length() != 0 && ((sub.length() - tmp.length()) % 2) == 0) {
                        // 出现偶数次
                        //System.out.println(i +"-->"+j +":"+sub);
                        if (maxSub.length() <= sub.length()) {
                            maxSub = sub;
                        }
                    }
                }
            }
            //System.out.println("---->"+maxSub);

            return maxSub.length();
        }
    }
```