一、编程题

ACM：最长子字符串的长度（二）

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'l'、'o'、'x' 字符都恰好出现了偶数次最长子字符串的长度。

输入描述：输入是一串小写的字母组成的字符串。

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 'l'，'o'各 2 个，以及 0 个 'x'。

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 'l' ，2个 'o' ，2个 'x'。

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 'l'、'o'、'x' 都出现了 0 次。

代码：

```
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 line = in.nextLine();
            if (line.length() == 0) {
                System.out.println(0);
                continue;
            }
            int result = 0;
            for (int i = 0; i < line.length(); i++) {
                result = Math.max(getResult(line, i), result);
            }
//              int result1 = getResult(line, 3);
//              System.out.println("result1 = " + result1);
            System.out.println(result);
        }
    }

    public static int getResult(String line, int target) {
//          System.out.println(line);
```

```
            int[] dp = new int[line.length()];
            int result = 0;
            int start = target == line.length() - 1 ? 0 : target + 1;
            int l_count = 0;
            int o_count = 0;
            int x_count = 0;
            int i = 0;
            char c = line.charAt(target);
            if (c == 'l') {
                  l_count++;
            } else if (c == 'o') {
                  o_count++;
            } else if (c == 'x') {
                  x_count++;
            }
            dp[0] = isValid(l_count, o_count, x_count) ? 1 : 0;
            i++;
            while (start != target) {
                  c = line.charAt(start);
                  if (c == 'l') {
                        l_count++;
                  } else if (c == 'o') {
                        o_count++;
                  } else if (c == 'x') {
                        x_count++;
                  }
                  if (isValid(l_count, o_count, x_count)) {
//                        System.out.println("start = " + start);

                        if (start >= target) {
                              dp[i] = start - target + 1;
                        } else {
                              dp[i] = line.length() - (target - start) + 1;
                        }
                  } else {
                        dp[i] = dp[i - 1];
                  }
                  if (start + 1 >= line.length()) {
                        start = 0;
                  } else {
                        start ++;
                  }

                  result = Math.max(result, dp[i]);
```

```
//                    System.out.println("result = " + result);
                    i++;
            }
//              System.out.println(Arrays.toString(dp));
            return result;
      }

      public static boolean isValid(int a, int b, int c) {
            return a % 2 == 0 && b % 2 == 0 && c % 2 == 0;
      }
}
```