相对开音节

题目描述：

相对开音节构成的结构为辅音+元音（aeiou）+辅音(r除外)+e，常见的单词有bike、cake等。

给定一个字符串，以空格为分隔符，反转每个单词中的字母，若单词中包含如数字等其他非字母时不进行反转。

反转后计算其中含有相对开音节结构的子串个数（连续的子串中部分字符可以重复）。

输入描述：

字符串，以空格分割的多个单词，字符串长度<10000，字母只考虑小写

输出描述：

含有相对开音节结构的子串个数，注：个数<10000

补充说明：

示例1

输入：

```
ekam a ekac
```

输出：

```
2
```

说明：

反转后为 make a cake 其中make、cake为相对开音节子串，返回2

示例2

输入：

```
!ekam a ekekac
```

输出：

```
2
```

说明：

反转后为!ekam a cakeke 因!ekam 含非英文字符所以未反转，其中 cake、keke 为相对开音节子串，返回 2

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


// 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.hasNext()) { // 注意 while 处理多个 case
            String line = in.nextLine();
            String[] g = line.split(" ");
            int n = g.length;
            int result = 0;
            Pattern other = Pattern.compile("[^a-z]");
            Pattern target = Pattern.compile("[^aeiou][aeiou][^aeiour]e");
            Pattern reverse = Pattern.compile("e[^aeiour][aeiou][^aeiou]");
            for (int i = 0; i < n; i++) {
                int bound = g[i].length() - 4 + 1;
                Matcher matcher = other.matcher(g[i]);
```

```
                boolean flag = matcher.find();
                for (int j = 0; j < bound; j++) {
                    String s = g[i].substring(j, j + 4);
                    if (other.matcher(s).find()) continue;
                    if (flag) {
                        if (target.matcher(s).find()) {
                            result++;
                        }
                    } else {
                        if (reverse.matcher(s).find()) {
                            result++;
                        }
                    }
                }
            }
            System.out.println(result);
        }
    }
}
```