Pre Project2

前提说明

1. 本系列作业数据范围规定为：总输入字符数 不超过150000个且单词最大长度不超过100
2. 输入输出方式为标准输入输出，即控制台方式，stdin & stdout
3. 输入数据以 EOF 结尾，即不保证以 '\n' 结尾
4. 本单元作业原则上不要求考虑输入错误及异常处理的情况，即 保证测试数据正确的正确性
5. 本单元作业同样是以 Task 的演进方式进行迭代，虽然我们直接给了同学们从 Task1 ~ Task6 的所有题面，但还是希望同学们能够从 Task1 开始顺序做题，并在做题的同时思考如何给后续的迭代留下修改的空间，而不至于同一个单元作业中每一个新的 Task 都要重新写一份代码。这需要同学们运用Java中面向对象的思想而不是C语言中面向过程的解题方式。Pre-project 的作业需要珍惜，因为开学过后的正式OO课程中的作业难度以及代码量将会是寒假作业的几倍，重构和重写将会变得十分繁重， 所以希望大家在动键盘前多思考，培养面向对象的思想，在后面的正式课程中才能游刃有余。

Task1：

描述：

输入一行英文句子，输出其单词个数

输入：

保证输入的字符中只可能出现以下字符：

1. 大小写字母
2. 逗号 ','
3. 句号 '.'
4. 回车 '\n'
5. 空格 ' '

单词定义为用 标点和空格 分开的一串连续的只包含字母的字符串

即两个单词之间只有可能出现 三种 分隔方式：

1、空格

2、逗号+空格

3、句号+空格

输出：

输出一行一个数，表示单词个数

样例输入：

```
ni hao, ni jiu shi da lao.
```

样例输出：

```
7
```

Task2:

描述

输入一篇英语文章，由多行组成，提取出其中的所有单词，重复的单词只输出一次。

输入：

保证输入的字符中只可能出现以下字符：

1. 大小写字母
2. 逗号 ','
3. 句号 '.'
4. 回车 '\n'
5. 空格 ' '

单词定义为用 标点和空格和回车 分开的一串连续的只包含字母的字符串

即两个单词之间只有可能出现 四种 分隔方式：

1、空格

2、逗号+空格/回车

3、句号+空格/回车

4、回车

输出：

将每个单词转成全小写，并按字典序输出，每个单词输出一行，

样例输入：

```
Hello. My dear
friends. nice to meet u.
```

样例输出：

```
dear
friends
hello
meet
my
nice
to
u
```

Task3:

描述：

在上一次的基础上，输入增加连接符，并实现词频统计

输入：

保证输入的字符中只可能出现以下字符：

1. 大小写字母
2. '-'连接符（减号）(new!)
3. 逗号','

4. 句号'.'
5. 回车'\n'
6. 空格' '

连接符的说明：

1. 连接符 可以 出现在一行一个单词中间，如：post-graduate，这样算作一个单词(不忽略'-'，具体请看例子)而不是两个分开的单词
2. 连接符 不会 出现在单词的 第一个字符或最后一个字符 ，即 保证输入中 不存在形如 -all 或者 post- 的单词
3. 忽略连接符的情况：连接符 也可以 出现在行末，用来连接下一行的第一个字符串（保证输入中 下一行只会以字母开头或为空），并视为 一个单词 ，统计此单词时应 忽略 此连接符
4. 每个单词中最多出现一个连接符(第一种连接符)，即统计后单词表中不存在形如aaa-bbb-ccc-ddd..的单词

   ```
   dear fri-
   end:
   Real-Time
   Real-Ti-
   me test
   ```

   时，正确统计结果为：

   dear 出现一次

   friend 出现一次

   real-time 出现两次

   test 出现一次

单词定义为用 标点和空格和回车 分开的一串连续的由字母和连接符组成的字符串

输出：

将每个单词转成全小写，并按 字典序 输出单词和词频

每行输出一个单词的信息，格式为

```
单词 (空格) 单词出现的次数 (空格) 该单词出现的次数占总单词的数量之比
```

注：最后一个比值为四舍五入保留两位小数的百分数

样例输入：

```
ni hao,
Ni shi zui pang de.
```

样例输出：

```
de 1 14.29%
hao 1 14.29%
ni 2 28.57%
pang 1 14.29%
shi 1 14.29%
zui 1 14.29%
```

Task4:

描述：

在上一次的基础上，输入增加感叹号和问号，并实现单词出现位置统计

输入：

保证输入的字符中只出现以下字符：

1. 大小写字母
2. '-'连接符（减号）
3. 逗号','
4. 句号'.'
5. 感叹号'!'(半角英文感叹号) (new!)
6. 问号'?'(半角英文问号) (new!)
7. 回车'\n'
8. 空格' '

连接符定义方式同上一个Task

单词定义为用 标点和空格和回车 分开的一串连续字符串

即两个单词之间只有可能出现 六种 分隔方式：

1、空格

2、逗号+空格/回车

3、句号+空格/回车

4、感叹号+空格/回车 (new!)

5、问号+空格/回车 (new!)

6、回车

输出：

将每个单词转成全小写，并按字典序输出单词和词频以及出现位置

每n+1行输出一个单词的信息，n为该单词出现次数，格式为

```
单词 (空格) 单词出现的次数 (空格) 该单词出现的次数占总单词的数量之比
(制表符) (r1, c1)
(制表符) (r2, c2)
(制表符) (r3, c3)
   ......
```

注：

1. 最后一个比值为四舍五入保留两位小数的百分数
2. r_i 和 c_i 分别为该单词 第i次 出现的 行数 和 在该行的起始位置 ，每个坐标前面为制表符 '\t'
3. 出现位置按先后顺序输出
4. 从1开始计数，即不存在第0的说法

样例输入：

```
ni hao,
ni shi zui pang de.
```

样例输出：

```
de 1 14.29%
   (2, 17)
hao 1 14.29%
   (1, 4)
ni 2 28.57%
   (1, 1)
   (2, 1)
pang 1 14.29%
   (2, 12)
shi 1 14.29%
   (2, 4)
zui 1 14.29%
   (2, 8)
```

Task5:

描述：

在上一个task的基础上，定义一个单词的翻转为把它的所有字符翻转过来，如 rev(pang)=gnap

我们认为一个单词和它的翻转是同一个单词，在这基础上实现Task4的词频统计，输出单词是输出它和它的翻转中字典序更小的那个。

输入：

输入方式同上一个Task

样例输入

```
ni
In
```

输出：

输出方式同上一个Task

样例输出

```
in 2 100.00%
  (1, 1)
  (2, 1)
```

task6:

描述：

在task4的基础上(即不考虑翻转)，完成对单词的分类，且本题需要区分大小写，同时允许单词内出现多个不相邻的连接符。

子串：字符串中任意个连续的字符组成的子序列称为该串的子串

子序列：在原来序列中找出一部分组成的序列，一个字符串abc的子序列有a, b, c, ab, ac, bc, abc

前缀：包含字符串第一个字符的子串

后缀：包含字符串最后一个字符的子串

缩写：

- a(x)表示连续的x个a, 如a(3) = aaa
- ab(x)表示连续的x个ab，如ab(3) = ababab

+：字符串的拼接，aa + bb = aabb

lower(s)：将s的所有字符转为小写，lower(ABcc) = abcc

我们定义如下几类单词：

- A类：
  - 单词中存在一个子串为a(x) + b(y) + a(z) + c(i)
  - x∈[2, 3], y∈[2,4], z∈[2, 4], i∈[2, 3]
  - 例如aabbaacc
- B类：
  - 单词中存在一个子串为a(x) + ba(y) + bc(z)
  - x∈[2, 3], y∈[0,100000000], z∈[2, 4]
  - 例如aabcbcbc, aababcbc
- C类:

    - 单词中存在一个子串s满足lower(s) = a(x) + ba(y) + bc(z)
    - x∈[2, 3], y∈[0,100000000], z∈[2, 4]
    - 例如AAbcBc
- D类：
    - 该单词存在一个前缀为a(x) + b(y) + c(z)
    - x∈[0, 3], y∈[1,1000000], z∈[2, 3]
    - 且该单词存在一个后缀s满足lower(s) = b(i) + a(j) + c(k)
    - i∈[1, 2], j∈[1,2], k∈[0, 3]
    - 例如bccxxbA，aaabccxxbAcCc，但是xaabx，bcc不属于此类
- E类:
    - 该单词存在一个子序列a(x)+b(y)+c(z)+b(i)+c(j)
    - x∈[1, 3], y∈[2, 10000], z∈[1, 2]
    - i∈[1, 3], j∈[2, 10000]
    - 例如axaxbxbxcxbxcxc

输入：

输入方式同Task4

样例输入

```
aabbaacc
aabcbcbc
AAbcBc
bccxxbA
axaxbxbxcxbxcxc
y
```

输出：

先把单词按字典序排序，对于每个单词，输出一行(同一个单词出现多次只输出一次)

每行第一个数k表示它属于多少个类，接下来k个字符表示它所属的类别，数k和k个字符之间用空格分隔。

样例输出

```
1 C
1 A
2 BC
1 E
1 D
0
```