Language detection

You are given a corpus of text documents written in different languages. The goal is to create a software component which will automatically detect the language of a given text in the form of a paragraph, sentence, word, or a few letters.

1. Count the number of appearances of letter bigrams in the text for each language Li.
   - Letter bigrams that contain new line characters (\r and \n) should be ignored.
   - All other letter bigrams should be counted, including letter bigrams that contain white space, punctuation marks...
   - Bigrams should be case-insensitive, e.g. ab, aB, Ab, AB should be counted as occurrences of the same, lowercase bigram ab.
2. Based on results from 1. find the probabilities that a language Li is used to write a given text sequence text, i.e. find P(Li|text). Consider that with no prior information all languages are equally likely, i.e. that prior language probabilities P(Li) are equal.

Hints

- Based on letter bigram appearances counted in 1. find the probability that a bigram xy appears in language Li, P(xy|Li). P(text|Li) should be approximated using a letter bigram model, i.e. text should be modeled as a sequence of conditionally independent letter bigrams, given a language. For additional details refer to the following wikipedia page.
- Based on P(text|Li) calculate probabilities that a language Li is used to write a given text sequence text, i.e. find P(Li|text). For more details refer to Bayes' theorem.

Data sets

There are two data sets:

- *Public data set* is used for developing your solution. After you submit your solution, you will be able to see how well your solution performs against this data set. *Public data set* is not used for calculating the final score. Public data set is available **here**.
- *Private data set* is used for testing your solution. The final score will be measured against this data set. *Private data set* and the final score will be available after the homework finishes. *Private data set* contains different data than the *public data set*, but the type of data (text length, number of text files, depth of folder hierarchy...) is roughly the same. Private data set is available **here**.

Input

Inputs are given through standard input.

Path to the folder which contains corpus of plain text documents is given as the first input line. Text files written in the same language can be found in hierarchy of the same sub-folder in the corpus folder. One such layout is depicted below.

```
<corpus_folder>/
 ├───<L0>/
|   ├───<f0_0>.txt
|   ├───<f0_1>.txt
|   └───<f0_2>.txt
 ├───<L1>/
|   ├───<f1_0>.txt
|   ├───<d1_0>/
|   |   ├───<f1_0_0>.txt
|   |   └───<f1_0_1>.txt
|   └───<d1_1>/
|       └───<f1_1_0>.txt
 └───<L2>/
    └───<d2_0>/
        └───<d2_0_0>/
            └───<f2_0_0_0>.txt
```

Text corpus

Names of folders <L0>, <L1>, <L2> are two-letter (639-1) language nomenclatures, e.g. en for English, fr for French, de for German. There will be two or more language sub-folders. For more details about two-letter (639-1) language nomenclatures refer to the following wikipedia page. Names of the rest of the folders and *.txt files in hierarchy can take arbitrary values. There will be at least one *.txt file in hierarchy of each language sub-folder.

Each *.txt file is encoded using UTF-8 encoding.

Input text sequences

Path to the file which contains input text sequences for tasks 2. will be given as the second input line. The file will not be empty and will not contain empty line. Input text sequences are arranged one per line, e.g.

```
<input_0>
<input_1>
<input_2>
```

Input text file is encoded using UTF-8 encoding.

Input text sequences can contain leading and/or trailing spaces.

Input text sequences are not empty.

Output

All results should be printed to the standard output. Results for task 1 should be printed first, followed by results for task 2. There should be no empty lines separating results of different tasks.

1. Resulting list of letter bigrams and their appearances per language should be printed to the standard output. Single output line should consist of a comma-separated triplet (Li, bigram, count). You should print 5 most frequent

bigrams per language, sorted by occurrences, descending. Bigrams with the same frequencies should be lexically sorted. Languages should be sorted lexicographically, ascending, according to their two-letter nomenclature. Example of the output:
<L0>,<c_000><c_001>,<count_00>

2. <L0>,<c_010><c_011>,<count_01>
3. <L0>,<c_020><c_021>,<count_02>
4. <L0>,<c_030><c_031>,<count_03>
5. <L0>,<c_040><c_041>,<count_04>
6. <L1>,<c_100><c_101>,<count_10>
7. <L1>,<c_110><c_111>,<count_11>
8. <L1>,<c_120><c_121>,<count_12>
9. <L1>,<c_130><c_131>,<count_13>
10. <L1>,<c_140><c_141>,<count_14>
11. <L2>,<c_200><c_201>,<count_20>
12. <L2>,<c_210><c_211>,<count_21>
13. <L2>,<c_220><c_221>,<count_22>
14. <L2>,<c_230><c_231>,<count_23>
15. <L2>,<c_240><c_241>,<count_24>
16.

    where <Li> is two-letter language nomenclature, <c_ijk><cijl> is the bigram and count_im is is the number of occurrences of bigram <c_ijk><cijl> in language <Li>.
17. For each input text sequence <input_j>, print the probability that the language Li is used to write the sequence. Probabilities should be printed one per line, sorted by lexicographical order of language nomenclatures, ascending. Example:
    <L0>,P(L0|input_0)
18. <L1>,P(L1|input_0)
19. <L2>,P(L2|input_0)
20. <L0>,P(L0|input_1)
21. <L1>,P(L1|input_1)
22. <L2>,P(L2|input_1)
23. <L0>,P(L0|input_2)
24. <L1>,P(L1|input_2)
25. <L2>,P(L2|input_2)
26.

    where <Li> is two-letter language nomenclature, and P(Li|input_j) is the probability that text sequence input_j is written in the language Li.

Scoring

- Correct result for task 1 brings 40 points per test case.
- Correct result for task 2 brings 40 points per test case.
- Probabilities in task 2 are considered correct if they don't differ from the

expected values by more than 0.01.

Constraints

- Time limit is 2s.
- Memory limit is 64 MB.

If in doubt, please refer to the data from *public data set* and proceed with a reasonable assumption.