TF-IDF

You are given the task to go through a corpus of text documents in English, analyze their content and give a short presentation on the most important takeaways. You quickly realized that there is a large number of documents and that you cannot do this task manually. Luckily, you found an awesome technique to do a statistical analysis of the words in files called TF-IDF. In order to be able to quickly go through the data, you decided to write a software component which will be able to extract the 10 most important terms in a document and provide a short, 5 sentence summary.

TF-IDF statistic

Intuitively, a term is more important if it occurs more frequently in a document. However, some common words will appear frequently in nearly all documents in a corpus. Therefore, the TF-IDF metric has two parts - term and document frequency.

The number of times a term occurs in a document is called its **term frequency**. Each term is a single stem of a word, and two words are considered to be the same term if their stems match. For example, "think" and "thinking" are considered the same term because their stem is "think", but "rethink" and "thinker" are not. Libraries that remove morphological affixes from words, leaving only the word stem, are called stemmers. For the purposes of this problem, you should use the `SnowballStemmer` from the NLTK library.

Document frequency is defined as the number of documents in a corpus that contain at least one occurence of the given term. The inverse document frequency is then calculated as `idf(t) = log(N/k(t))`, where `N` is the number of documents in the corpus, and `k(t)` the number of documents that contain the term t. (Defining the metric in the special case when `k(t) = 0` will not be needed for this problem.)

Finally, The TF-IDF metric of a term in a document that's part of a corpus is obtained by multiplying its TF and IDF metrics.

Problem statement

For each document, you are given two tasks.

1. Find the 10 terms in the given document with the largest TF-IDF score, in order of descending score. In order to find the top 10 terms, you will need to first tokenize the words in the documents and process them by removing morphological affixes. If the document contains less than 10 different terms, print all the terms as stated above. Ignore terms that contain non-alphanumerical characters (use check `term.isalnum()`).

2. Extract the summary of a document using a technique based on TF-IDF. For each sentence, the sentence relevance score is defined as the sum of the TF-IDF scores of the most important 10 words (that are not necessarily distinct from each other) in the sentence. The summary is then obtained by taking the 5 sentences with the greatest relevance scores, in the same order in which they appear in the original document. (If a sentence has fewer than 10 words, its sentence score is simply the sum of TF-IDF scores of all its words. If the document consists of 5 or fewer sentences, the summary is simply the entire document.)

Here is the code snippet for importing the libraries you will use for this purposes:

```
 # import these modules
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize, sent_tokenize
```

Your code should take path to the document as an input and generate a two line output, as specified in the instructions below. Due to Petlja's infrastructure limitations, you should add this code to your solution before printing the output, in order for your results to be stored properly:

```
 import sys
sys.stdout.reconfigure(encoding='utf-8')
```

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. The data in the *private data set* is different form the data in the *public data set*, but the type of data (text length, number of text files, depth of folder hierarchy...) is roughly the same.

Input

Inputs are given through standard input.

The input will have exactly two lines. The first input line contains the path to the folder which contains the corpus of documents. The second input line contains the path to the `*.txt` document that needs to be analyzed.

Within the folder, there can be multiple subfolders, each containing one or more `*.txt` files, which are all part of the corpus. An example is given below.

```
<corpus_folder>/
├──<subfolder_1>/
│   ├──<f0>.txt
│   ├──<f1>.txt
│   └──<f2>.txt
├──<subfolder_2>/
│   ├──<f3>.txt
│   ├──<subfolder_3>/
│   │   ├──<f4>.txt
│   │   └──<f5>.txt
│   └──<subfolder_4>/
│       └──<f5>.txt
└──<subfolder_5>/
    └──<f6>.txt/
        └──<subfolder_6>/
            └──<f7>.txt
```

The names of all folders and `*.txt` files in hierarchy can take arbitrary values. There will be at least one `*.txt` file in the hierarchy of each subfolder.

Each `*.txt` 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 each document should be printed in form:

```
 <task_1_output>
<task_2_output>
```

The result of Task 1 should be printed in the first line, followed by the results for Task 2 in the second line. There should be no empty lines separating results of different tasks.

1. For the first task, the resulting list of top 10 terms should be printed to the standard output in a single line that consists of comma-separated terms (`"term1, term2, ..., term10"`). You should print these terms in the order given by the TF-IDF score, from most to least significant. When multiple words have the same score, you should order them lexicographically.

2. For the second task, the resulting list of 5 sentences should be printed to the standard output in a single line. The sentences should be separated by a single space (`"Sentence1. Sentence2. ... SentenceN."`). The order of the sentences should be the same as in the original document. If multiple sentences have the same score, the higher priority should be assigned to the sentence that comes earlier in the document.

Scoring

- Correct result for task 1 brings 50 points per test case.
- Correct result for task 2 brings 50 points per test case.

Constraints

- Time limit is 12s.
- Memory limit is 128 MB.

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