IEEEXplore Indexing

Time limit: *1000 ms*
Memory limit: *256 MB*

Warm greetings to all IEEEXtreme Participants from the Xplore API Team!

As part of the IEEEXtreme competition, in this challenge you are asked to identify the most important index terms of an academic journal. This challenge emulates a real-world application - The IEEE Xplore API uses a similar approach!

For a full dynamic database search IEEE Xplore API is available for your IEEE research needs. Xplore API provides metadata on 4.9M academic works and is now delivering full-text content on 50K *Open Access* articles. Xplore API will meet your research needs fast and easy. The Xplore API Portal supports PHP, Python and Java as well as providing output in Json and XML formats. Many API use cases are listed within the API Portal.

Xplore API registration is free. To learn more about IEEE Xplore API please visit developer.ieee.org and register for an API key TODAY!

Challenge

An index term, subject term, subject heading, or descriptor, in information retrieval, is a term that captures the essence of the topic of a document. Index terms make up a controlled vocabulary for use in bibliographic records. They are an integral part of bibliographic control, which is the function by which libraries collect, organize and disseminate documents.

In this challenge you are given a list of index terms, a list of stop words, and a document. You are asked to parse the document and identify the top 3 index terms that have the highest keyword density.

Definitions

For the purpose of this challenge:

- A *word* is a consecutive sequence of lowercase letters (`a-z`) or apostrophe (`'`). A word must not contain any other characters. The sequence must have at least 4 characters.
- A *punctuation* is a comma (`,`), a period (`.`), a question mark (`?`), or an exclamation mark (`!`).
- A *token* is a consecutive sequence of lowercase or uppercase letters (`a-zA-Z`), apostrophes (`'`), or punctuation. A token must not contain spaces or newlines.
- A *index term* is a word. A *stop word* is a word.
- A *document* is a simplified XML with matched *tags* such as `<body>` , and `</abstract>` . An opening tag has the format `<[a-z]+>` . A closing tag has the format `</[a-z]+>` . In other words, a tag only has lowercase letters in them and is always enclosed by a pair of brackets `<>` .
- There are three *special tags*: `<title>` , `<abstract>` , and `<body>` .
- Tags can be nested, such as `<a><p></p></a>` .
- A document may have an arbitrary number of tokens, spaces, or newlines between its tags. No tokens appear outside all the tags.

Keyword Density

Here is the methodology to compute the keyword density for an index term w:

- Each token in the document is normalized to its underlying word, by first converting all uppercase letters to lowercase, and then removing all punctuation in it.
- All words that are *stop words* are ignored.
- Compute the total number of words in the special tags of the document as L. A same word may appear multiple times in a special tag and is counted multiple times in L. Note that words outside special tags are not counted towards L.
- Compute the index term score S_w by its occurrences in the special tags: Each occurrence in the `<title>` scores 5. Each occurrence in the `<abstract>` scores 3. Each occurrence in the `<body>` scores 1.
- The *keyword density* of an index term w is defined as $S_w / L \cdot 100$

Standard input

The first line of the input has a list of stop words separated by a single semicolon.

The second line of the input has a list of index terms separated by a single semicolon.

From line three to the end of the input file gives the XML document.

Standard output

Output the top 3 index terms with the highest keyword density. Output each index term along with its keyword densities on a single line, with a colon and a space between them (see the sample). The index terms should be sorted by decreasing keyword density. Your keyword density is considered correct if it has an absolute or relative error of at most 10^{-6} from the correct keyword density.

Output only the index terms with a positive keyword density. If there are fewer than 3 index terms that appear in the document, output only those that appear.

In case there is a tie on keyword densities, all index terms that have a keyword density that is as large as the 3rd highest keyword density should be printed. If two index terms have a same keyword density, the lexicographically smaller word should be printed first.

Constraints and notes

- The number of index terms is between 5 and 30.
- The number of stop words is between 5 and 150.
- All index terms are distinct. All stop words are distinct. No index term is a stop word, and vice versa.
- The document contains exactly one `<title>` , one `<abstract>` , and one `<body>` tag. No special tag is in another special tag. However, a special tag can be nested in other tags.
- The document contains only lowercase letters (`a-z`), uppercase letters (`A-Z`), apostrophes (`'`), punctuation (`,.?!`), XML tags, spaces (not tabs), or newlines.
- The document contains at most $20\,000$ characters.
- No line of the document has trailing spaces. However, a line may have leading spaces as indentation.
- No line of the document contains only the newline character. There are no empty lines.
- Each XML tag starts and ends on a same line.
- The input files of this challenge are not only chosen from real-world scenarios, but may also be specially constructed to test that your computation is correct, just as in the other challenges.

Input

```
being;does;have;haven't;more;should;shouldn't;than;that;these;wha
t
classification;cryptography;diseases;probability;stability
<response>
 <article>
  <title>A Novel Approach to Image Classification, in a Cloud Com
puting Environment stability.</title>
  <publicationtitle>IEEE Transactions on Cloud Computing</publica
tiontitle>
  <abstract>Classification of items within PDF documents has alwa
ys been challenging.  This stability document will discuss a simp
le classification algorithm for indexing images within a PDF.</ab
stract>
 </article>
 <body>
  <sec>
   <label>I.</label>
   <p>Should Haven't That is a bunch of text pattern these classi
fication and cyrptography.  These paragraphs are nothing but nons
ense.  What is the statbility of your program to find neural nets
. Throw in some numbers to see if you get the word count correct
this is a classification this in my nd and rd words.  What the he
ck throw in cryptography.</p>
   <p>I bet diseases you can't find probability twice.  Here it i
s a again probability.  Just to fool you I added it three times p
robability.  Does this make any pattern classification? pattern c
lassification! pattern classification.</p>
   <p>
    <fig>
     <label>FIGURE.</label>
     <caption>This is a figure representing convolutional neural
nets.</caption>
    </fig>
   </p>
 </sec>
 </body>
</response>
```

Output

```
classification: 19.512195122
stability: 9.756097561
probability: 3.658536585
```

Explanation

There are a total of 82 words. Their scores S_w are:

- classification: 16
- stability: 8
- probability: 3
- cryptography: 1
- diseases: 1

For `classification`, the keyword density is $16/82 \cdot 100 \approx 19.512195$. The other keyword densities can be computed similarly.

Input

```
what;when;where;like;that
welcome;ieee;xtreme;ieeextreme;programming
<title>Welcome to IEEEXtreme!</title>
<keyword>welcome, ieeextreme</keyword>
<abstract>
Welcome!Participants!!!
IEEE Xtreme is a global challenge in which teams of IEEE Student
members, advised and proctored by an IEEE member, and often suppo
rted by an IEEE Student Branch.
Compete in a twentyfour hour timespan against each other to solve
 a set of ...XtremE... programming problems in IEEEXtreme.
</abstract>
<body>WELCOME. wel.come... Are you ready? Good luck and have fun
in xtreme!</body>
<other>
Mark your calender and don't miss the action.
</other>
```

Output

```
ieee: 30.000000000
ieeextreme: 20.000000000
welcome: 17.500000000
xtreme: 17.500000000
```

Explanation

Make sure that you read the problem statement correctly. Do not forget to handle the tied index terms and output them in lexicographical order.