1071. Speech Patterns (25)

时间限制

300 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

HOU, Qiming

People often have a preference among synonyms of the same word. For example, some may prefer "the police", while others may prefer "the cops". Analyzing such patterns can help to narrow down a speaker's identity, which is useful when validating, for example, whether it's still the same person behind an online avatar.

Now given a paragraph of text sampled from someone's speech, can you find the person's most commonly used word?

**Input Specification:**

Each input file contains one test case. For each case, there is one line of text no more than 1048576 characters in length, terminated by a carriage return '\n'. The input contains at least one alphanumerical character, i.e., one character from the set [0-9 A-Z a-z].

**Output Specification:**

For each test case, print in one line the most commonly occurring word in the input text, followed by a space and the number of times it has occurred in the input. If there are more than one such words, print the lexicographically smallest one. The word should be printed in all lower case. Here a "word" is defined as a continuous sequence of alphanumerical characters separated by non-alphanumerical characters or the line beginning/end.

Note that words are case **insensitive**.

**Sample Input:**

Can1: "Can a can can a can? It can!"

**Sample Output:**

can 5

[提交代码](https://www.patest.cn/contests/pat-a-practise/1071)

有了map以后查找之类的操作就变得很轻松了，这里还有一个特判，就是当字符串没有终止符的时候将不会在map里放入任何字符，这个时候只需要输出原句+1就好了。

#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

#include<map>

#include<cstring>

#include<climits>

using namespace std;

int check(char target)

{

if (target >= 'a'&&target <= 'z' || target <= 'Z'&&target >= 'A' || target >= '0'&&target <= '9')

return 1;

else

return 0;

}

map<string, int>store;

string sequence;

int main()

{

getline(cin, sequence);

string temp;

int flag = 0;

for (int i = 0; i < sequence.size(); i++)

{

if (sequence[i] <= 'Z'&&sequence[i] >= 'A')

{

sequence[i] -= 'A' - 'a';

}

if (check(sequence[i]))

{

flag = 1;

temp.push\_back(sequence[i]);

continue;

}

else if(flag==1)

{

map<string, int>::iterator it = store.find(temp);

if (it == store.end())

{

store[temp] = 1;

}

else

{

it->second++;

}

temp.clear();

flag = 0;

}

}

map<string, int>::iterator it;

map<string, int>::iterator max;

int max\_num=-1;

for (it = store.begin(); it != store.end(); it++)

{

if (it->second > max\_num)

{

max = it;

max\_num = it->second;

}

}

if(store.size())

cout << max->first << " " << max->second;

else

{

cout << sequence << " " << '1';

}

}