题目描述：

给定一个连续不包含空格字符串，该字符串仅包含英文小写字母及英文文标点符号(逗号、分号、句号)，同时给定词库，对该字符串进行精确分词。
说明：
1.精确分词： 字符串分词后，不会出现重叠。即“ilovechina” ，不同词库可分割为 “i，love，china” “ilove，china”，不能分割出现重叠的"i，ilove，china",i 重叠出现
2.标点符号不成词，仅用于断句
3.词库：根据外部知识库统计出来的常用词汇例：
dictionary=["i","love","china","lovechina","ilove"],
4.分词原则：采用分词顺序优先且最长匹配原则
“ilovechina”，假设分词结果 [i,ilove,lo,love,ch,china,lovechina] 则输出[ilove，china]
错误输出：[i,lovechina], 原因："ilove ">优先于 "lovechina"成词
错误输出：[i,love,china] 原因："ilove" >"i" 遵循最长匹配原则

输入描述：

字符串长度限制：0<length<256
词库长度限制： 1<length<100000
第一行输入待分词语句 "ilovechina"
第二行输入中文词库 "i,love,china,ch,na,ve,lo,this,is,the,word"

输出描述：

按顺序输出分词结果 "i,love,china"

补充说明：

示例 1

输入：

```
ilovechina

i,love,china,ch,na,ve,lo,this,is,the,word
```

输出：

```
i,love,china
```

说明：

示例 2

输入：

```
iat

i,love,china,ch,na,ve,lo,this,is,the,word,beauti,tiful,ful
```

输出：

```
i,a,t
```

说明：

单个字母，不在词库中且不成词则直接输出单个字母

示例 3

输入：

```
ilovechina,thewordisbeautiful
i,love,china,ch,na,ve,lo,this,is,the,word,beauti,tiful,ful
```

输出：

```
i,love,china,the,word,is,beauti,ful
```

说明：

标点符号为英文标点符号

```
#include <iostream>
#include <vector>
#include <map>
#include <string>
#include <set>
#include <cstring>
using namespace std;

int get_map_max_value(map<int, int> &input){
    if(input.empty())
        return 1;
    int max_id = 0;
    int i = 0;
    for(auto iter: input){
        if (iter.second > max_id){
            max_id = iter.second;
        }
    }
    return max_id;
```

```
}

vector<string> get_match_ex(vector<string> &words_vec,vector<string> &dict){
    vector<string> res;
    for(auto words : words_vec){
        for(size_t i=0; i<words.length(); ){
            map<int, int> best_match_words;
            for(size_t j=0; j < dict.size(); ++j){
                if(strncmp(dict[j].c_str(),words.c_str() + i, dict[j].length())
== 0){
                    best_match_words[j] = dict[j].length();
                }

            }
            int index = get_map_max_value(best_match_words);

            std::string str = words.substr(i,index);
            res.emplace_back(str);
            i+= index;
        }
    }
    return res;
}



int main() {
    string words_str;
    vector<string> words;
    vector<string> dict;

    cin >> words_str;
    std::string dict_str;
    cin >> dict_str;
    int length = dict_str.length();
    std::string tmp_str;
    int index =dict_str.find(',');

    while(index != std::string::npos){
        tmp_str = dict_str.substr(0, index);
        dict.emplace_back(tmp_str);
        dict_str = dict_str.substr(index+1);
        index =dict_str.find(',');
    }
```

```
    if(!dict_str.empty()){
        dict.emplace_back(dict_str);
    }

    index =words_str.find(',');

    while(index != std::string::npos){
        tmp_str = words_str.substr(0, index);
        words.emplace_back(tmp_str);
        words_str = words_str.substr(index+1);
        index =words_str.find(',');
    }
    if(!words_str.empty()){
        words.emplace_back(words_str);
    }

    auto res = get_match_ex(words, dict);
    for(size_t i=0; i<res.size() -1; ++i){
        cout << res[i] << ",";
    }
    cout << res[res.size() -1] << endl;

}
// 64 位输出请用 printf("%lld")
```