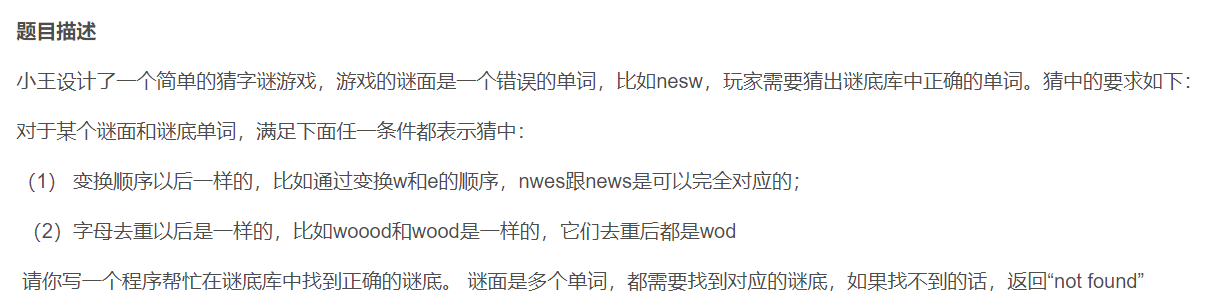
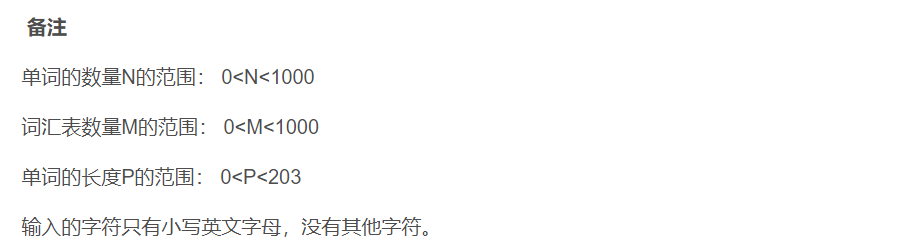
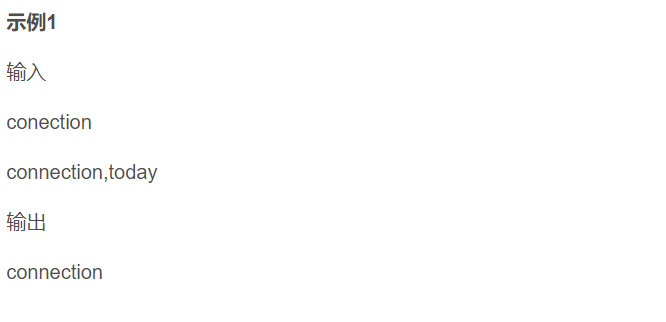
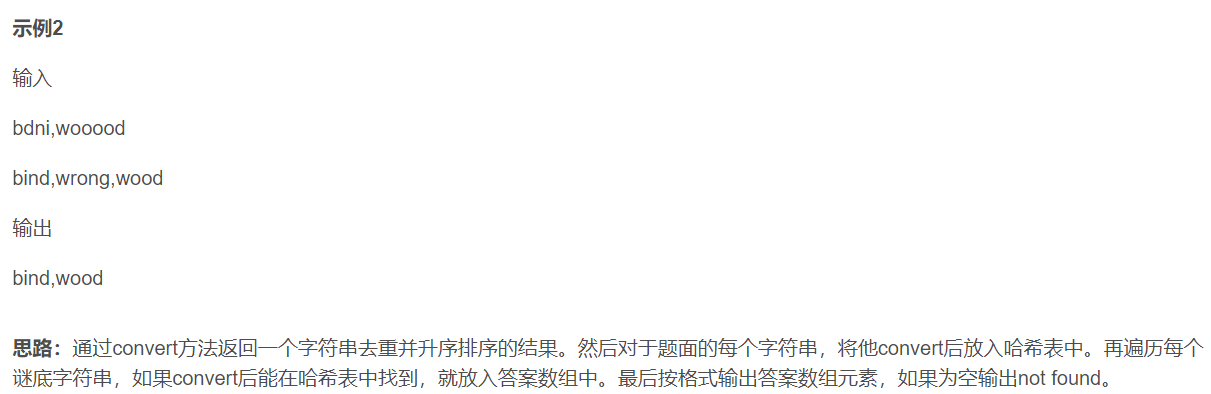
# **E卷-猜字谜[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**













import java.util.\*;

public class Main {

public static String convert(String s) {

char[] chars = s.toCharArray();

Arrays.sort(chars);

StringBuilder sb = new StringBuilder();

for (char c : chars) {

if (sb.length() == 0 || sb.charAt(sb.length() - 1) != c) {

sb.append(c);

}

}

return sb.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String inputMian = scanner.nextLine();

String inputDi = scanner.nextLine();

String[] mian = inputMian.split(",");

String[] di = inputDi.split(",");

Set<String> st = new HashSet<>();

for (String i : mian) {

st.add(convert(i));

}

List<String> res = new ArrayList<>();

for (String i : di) {

String now = convert(i);

if (st.contains(now)) {

res.add(i);

}

}

if (res.isEmpty()) {

System.out.println("not found");

} else {

System.out.println(String.join(",", res));

}

scanner.close();

}

}



mian = input().split(",")

di = input().split(",")

def convert(s):

return "".join(sorted(set(s)))

st = set()

for i in mian:

st.add(convert(i))

res = []

for i in di:

now = convert(i)

if now in st:

res.append(i)

if len(res):

print(",".join(res))

else:

print("not found")



#include <iostream>

#include <sstream>

#include <vector>

#include <unordered\_set>

#include <algorithm>

using namespace std;

string convert(string s) {

sort(s.begin(), s.end());

auto it = unique(s.begin(), s.end());

s.erase(it, s.end());

return s;

}

int main() {

string inputMian, inputDi;

getline(cin, inputMian);

getline(cin, inputDi);

istringstream ssMian(inputMian);

istringstream ssDi(inputDi);

vector<string> mian;

vector<string> di;

string token;

while (getline(ssMian, token, ',')) {

mian.push\_back(token);

}

while (getline(ssDi, token, ',')) {

di.push\_back(token);

}

unordered\_set<string> st;

for (const string& i : mian) {

st.insert(convert(i));

}

vector<string> res;

for (const string& i : di) {

string now = convert(i);

if (st.find(now) != st.end()) {

res.push\_back(i);

}

}

if (res.empty()) {

cout << "not found" << endl;

} else {

for (size\_t i = 0; i < res.size(); ++i) {

if (i > 0) {

cout << ",";

}

cout << res[i];

}

cout << endl;

}

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// 函数声明

char\* convert(const char\* s);

int compare\_chars(const void\* a, const void\* b);

int main() {

char mianInput[1000];

char diInput[1000];

// 读取第一行输入

fgets(mianInput, sizeof(mianInput), stdin);

// 读取第二行输入

fgets(diInput, sizeof(diInput), stdin);

// 去掉输入字符串末尾的换行符

mianInput[strcspn(mianInput, "\n")] = 0;

diInput[strcspn(diInput, "\n")] = 0;

// 用逗号分割输入字符串并存入数组

char\* mian[100];

int mianCount = 0;

char\* token = strtok(mianInput, ",");

while (token != NULL) {

mian[mianCount++] = token;

token = strtok(NULL, ",");

}

char\* di[100];

int diCount = 0;

token = strtok(diInput, ",");

while (token != NULL) {

di[diCount++] = token;

token = strtok(NULL, ",");

}

// 使用结构体构建简单集合

struct Set {

char\* converted[100];

int count;

} st = { .count = 0 };

// 将mian数组中的每个字符串转换并存入集合

for (int i = 0; i < mianCount; ++i) {

char\* transformed = convert(mian[i]);

int found = 0;

for (int j = 0; j < st.count; ++j) {

if (strcmp(st.converted[j], transformed) == 0) {

found = 1;

break;

}

}

if (!found) {

st.converted[st.count++] = transformed;

} else {

free(transformed);

}

}

// 存储结果的数组

char\* res[100];

int resCount = 0;

// 处理di数组中的每个字符串并检查是否在集合中

for (int i = 0; i < diCount; ++i) {

char\* transformed = convert(di[i]);

for (int j = 0; j < st.count; ++j) {

if (strcmp(st.converted[j], transformed) == 0) {

res[resCount++] = di[i];

break;

}

}

free(transformed);

}

// 输出结果

if (resCount == 0) {

printf("not found\n");

} else {

for (int i = 0; i < resCount; ++i) {

if (i > 0) {

printf(",");

}

printf("%s", res[i]);

}

printf("\n");

}

// 释放动态分配的内存

for (int i = 0; i < st.count; ++i) {

free(st.converted[i]);

}

return 0;

}

// 字符比较函数，用于qsort

int compare\_chars(const void\* a, const void\* b) {

return (\*(const char\*)a - \* (const char\*)b);

}

// 转换字符串，将字符排序并去重

char\* convert(const char\* s) {

size\_t len = strlen(s);

char\* chars = malloc(len + 1);

strcpy(chars, s);

// 对字符进行排序

qsort(chars, len, sizeof(char), compare\_chars);

// 去重并构建结果字符串

char\* result = malloc(len + 1);

int index = 0;

for (size\_t i = 0; i < len; ++i) {

if (i == 0 || chars[i] != chars[i - 1]) {

result[index++] = chars[i];

}

}

result[index] = '\0';

// 释放临时字符数组

free(chars);

return result;

}



const readline = require("readline");

// 创建接口来读取标准输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

// 用于存储输入的两个字符串数组

let mianInput = "";

let diInput = "";

// 读取第一行输入

rl.question("", (input1) => {

mianInput = input1;

// 读取第二行输入

rl.question("", (input2) => {

diInput = input2;

rl.close();

// 转换字符串的函数

function convert(s) {

// 将字符串转换成字符数组并排序

let chars = s.split("").sort();

// 使用Set去重

let uniqueChars = [...new Set(chars)];

// 连接字符数组并返回结果

return uniqueChars.join("");

}

// 将输入字符串按逗号分割成数组

let mian = mianInput.split(",");

let di = diInput.split(",");

// 使用Set存储转换后的字符串，集合保证了唯一性

let st = new Set();

mian.forEach((item) => {

st.add(convert(item));

});

// 存储结果的数组

let res = [];

di.forEach((item) => {

let now = convert(item);

if (st.has(now)) {

res.push(item);

}

});

// 根据结果数组是否为空输出相应的结果

if (res.length === 0) {

console.log("not found");

} else {

console.log(res.join(","));

}

});

});



package main

import (

"bufio"

"fmt"

"os"

"sort"

"strings"

)

// 转换字符串，将字符排序并去重

func convert(s string) string {

// 将字符串转换成字符切片

chars := []rune(s)

// 对字符切片进行排序

sort.Slice(chars, func(i, j int) bool {

return chars[i] < chars[j]

})

// 使用字符串构建器来去重和拼接字符

var sb strings.Builder

for i := 0; i < len(chars); i++ {

// 跳过重复字符

if i == 0 || chars[i] != chars[i-1] {

sb.WriteRune(chars[i])

}

}

return sb.String()

}

func main() {

// 创建带缓冲的读取器

reader := bufio.NewReader(os.Stdin)

// 读取第一行输入

mianInput, \_ := reader.ReadString('\n')

// 读取第二行输入

diInput, \_ := reader.ReadString('\n')

// 去掉输入字符串末尾的换行符

mianInput = strings.TrimSpace(mianInput)

diInput = strings.TrimSpace(diInput)

// 将输入字符串按逗号分割成数组

mian := strings.Split(mianInput, ",")

di := strings.Split(diInput, ",")

// 使用映射存储转换后的字符串

st := make(map[string]struct{})

for \_, item := range mian {

transformed := convert(item)

st[transformed] = struct{}{}

}

// 存储结果的切片

var res []string

for \_, item := range di {

transformed := convert(item)

if \_, exists := st[transformed]; exists {

res = append(res, item)

}

}

// 根据结果切片是否为空输出相应的结果

if len(res) == 0 {

fmt.Println("not found")

} else {

fmt.Println(strings.Join(res, ","))

}

}