



You can view this report online at : <https://www.hackerrank.com/x/tests/1752290/candidates/58098056/report>

| | |
|------------------|--------------------------------------------------------------------------|
| Full Name: | Instructor |
| Email: | aisha.batool@sse.habib.edu.pk |
| Test Name: | CS 101 - LW14 - Fall 23 |
| Taken On: | 18 Nov 2023 01:31:01 PKT |
| Time Taken: | 11 min 26 sec/ 180 min |
| Work Experience: | > 5 years |
| Invited by: | Aisha |
| Skills Score: | |
| Tags Score: | <div>CS 101 10/10</div> <div>CS101 10/10</div> <div>Python 3 10/10</div> |

100%

60/60

scored in **CS 101 - LW14 - Fall 23** in 11 min 26 sec on 18 Nov 2023 01:31:01 PKT

Recruiter/Team Comments:

No Comments.

| | Question Description | Time Taken | Score | Status |
|----|-----------------------------------|--------------|--------|--------|
| Q1 | Popular Author > Coding | 38 sec | 10/ 10 | ✓ |
| Q2 | Anagrams > Coding | 45 sec | 10/ 10 | ✓ |
| Q3 | Know your shopping cart! > Coding | 27 sec | 10/ 10 | ✓ |
| Q4 | Merge by Key > Coding | 6 min 35 sec | 10/ 10 | ✓ |
| Q5 | Count Words > Coding | 57 sec | 10/ 10 | ✓ |
| Q6 | Contacts > Coding | 23 sec | 10/ 10 | ✓ |

QUESTION 1

✓

Correct Answer

Score 10

Popular Author > Coding

QUESTION DESCRIPTION

Challenge
Write a function called `popular_author` that accepts a list of book dictionaries `b` as a parameter, and *returns* a string with the full name(s) of the most popular author(s) (sorted), separated by a comma if need be..

Note
Each dictionary object in the list will contain three keys, `'title'`, `'author'`, and `'year'`

Sample

```
>>> popular_author([{'title' : 'C++ How to Program' , 'author' : 'Harvey Deitel' , 'year' : '1994'},{'title' : 'Introduction to Algorithms' , 'author' : 'Charles E. Leiserson' , 'year' : '1989'},{'title' : 'C# for Programmers' , 'author' : 'Harvey Deitel' , 'year' : '2008'},{'title' : 'Code Complete' , 'author' : 'Steve McConnell' , 'year' : '1993'}])
```

Harvey Deitel

```
>>> popular_author([{'title' : 'Advanced Research in VLSI' , 'author' : 'Charles E. Leiserson' , 'year' : '1986'}, {'title' : 'C++ How to Program' , 'author' : 'Harvey Deitel' , 'year' : '1994'},{'title' : 'Introduction to Algorithms' , 'author' : 'Charles E. Leiserson' , 'year' : '1989'}, {'title' : 'C# for Programmers' , 'author' : 'Harvey Deitel' , 'year' : '2008'},{'title' : 'Code Complete' , 'author' : 'Steve McConnell' , 'year' : '1993'}])
```

Charles E. Leiserson, Harvey Deitel

Input/Output

Input consists of a list literal that HackerRank will read in as `b`, pass to your function, and then output the resulting string.

Constraints

- `b` will contain at least one book record.

INTERVIEWER GUIDELINES

```
def popular_author(b):
    dic = {}
    for l in b:
        dic[l["author"]] = dic.get(l["author"], 0) + 1

    dic2 = {}
    for key, value in dic.items():
        dic2[value] = dic2.get(value, [])+[key]

    ans = list(sorted(dic2[max(dic2.keys())]))

    return ", ".join(ans)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def popular_author(b):
2     dic = {}
3     for l in b:
4         dic[l["author"]] = dic.get(l["author"], 0) + 1
5
6     dic2 = {}
7     for key, value in dic.items():
8         dic2[value] = dic2.get(value, [])+[key]
9
10    ans = list(sorted(dic2[max(dic2.keys())]))
11
12    return ", ".join(ans)
13
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|------------|------------|-------------|-----------|-------|------------|-------------|
| Testcase 0 | Easy | Sample case | ✔ Success | 3 | 0.0168 sec | 9.54 KB |
| Testcase 1 | Easy | Sample case | ✔ Success | 3 | 0.0149 sec | 9.53 KB |
| Testcase 2 | Easy | Hidden case | ✔ Success | 4 | 0.0144 sec | 9.32 KB |

No Comments

QUESTION 2



Correct Answer

Score 10

Anagrams > Coding

QUESTION DESCRIPTION

Two words are anagrams if they contain exactly the same letters but in a different order. Write a function ***are_anagrams(word1, word2)*** that returns True if the two parameters (strings) are anagrams.

Note:

You cannot use the sort() or sorted() function.

Also, note that case, numbers, and special characters are ignored.

Sample

```
>> are_anagrams('John Keats', 'a jet honks')
True

>>> are_anagrams('Tom Marvolo Riddle', 'I am Lord Voldemort')
True

>>> are_anagrams('Mortein', 'Timer On 7!')
True

>>> are_anagrams('banana', 'ban')
False

>>> is_anagram('earrtt', 'heart')
False

>>> is_anagram('Debit card', 'Bad credit')
True

>>> is_anagram('Mixed', 'Missed')
False
```

As you see in the above examples:

1. a space ' ' should not be counted as a separate character, e.g., "John Keats" and "a jet honks" are anagrams
2. also notice in the above examples, anagrams are not case sensitive

INTERVIEWER GUIDELINES

```
def are_anagrams(word1, word2):
    word1 = word1.lower()
    word2 = word2.lower()
    dic1 = dict()
    for ch in word1:
        if ch.isalpha():
            if ch in dic1:
                dic1[ch] = dic1[ch] + 1
            else:
                dic1[ch] = 0
    dic2 = dict()
    for ch in word2:
        if ch.isalpha():
            if ch in dic2:
```

```

        dic2[ch] = dic2[ch] + 1
    else:
        dic2[ch] = 0
    return dic1==dic2

```

CANDIDATE ANSWER

Language used: **Python 3**

```

1
2 def are_anagrams(word1, word2):
3     word1 = word1.lower()
4     word2 = word2.lower()
5     dic1 = dict()
6     for ch in word1:
7         if ch.isalpha():
8             if ch in dic1:
9                 dic1[ch] = dic1[ch] + 1
10            else:
11                dic1[ch] = 0
12    dic2 = dict()
13    for ch in word2:
14        if ch.isalpha():
15            if ch in dic2:
16                dic2[ch] = dic2[ch] + 1
17            else:
18                dic2[ch] = 0
19    return dic1==dic2
20

```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|-------------|------------|-------------|-----------|-------|------------|-------------|
| Testcase 0 | Easy | Sample case | ✔ Success | 0.4 | 0.038 sec | 12.2 KB |
| Testcase 1 | Easy | Sample case | ✔ Success | 0.4 | 0.0285 sec | 12.1 KB |
| Testcase 3 | Easy | Hidden case | ✔ Success | 0.4 | 0.0644 sec | 12.2 KB |
| Testcase 4 | Easy | Sample case | ✔ Success | 0.4 | 0.0612 sec | 11.9 KB |
| Testcase 5 | Easy | Sample case | ✔ Success | 0.4 | 0.032 sec | 12.1 KB |
| Testcase 6 | Easy | Sample case | ✔ Success | 0.4 | 0.0561 sec | 12.1 KB |
| Testcase 7 | Easy | Sample case | ✔ Success | 0.4 | 0.0288 sec | 12.1 KB |
| Testcase 8 | Easy | Hidden case | ✔ Success | 0.4 | 0.0336 sec | 12 KB |
| Testcase 9 | Easy | Hidden case | ✔ Success | 0.4 | 0.0396 sec | 12.1 KB |
| Testcase 10 | Easy | Hidden case | ✔ Success | 0.4 | 0.0418 sec | 11.9 KB |
| Testcase 11 | Easy | Hidden case | ✔ Success | 0.4 | 0.0285 sec | 12.2 KB |
| Testcase 12 | Easy | Hidden case | ✔ Success | 0.4 | 0.037 sec | 11.9 KB |
| Testcase 13 | Easy | Hidden case | ✔ Success | 0.4 | 0.0411 sec | 12.2 KB |
| Testcase 14 | Easy | Sample case | ✔ Success | 0.4 | 0.0343 sec | 12 KB |
| Testcase 15 | Easy | Hidden case | ✔ Success | 0.4 | 0.0283 sec | 12.1 KB |
| Testcase 16 | Easy | Hidden case | ✔ Success | 0.4 | 0.0286 sec | 11.9 KB |
| Testcase 17 | Easy | Hidden case | ✔ Success | 0.4 | 0.0334 sec | 12 KB |
| Testcase 18 | Easv | Hidden case | ✔ Success | 0.4 | 0.0301 sec | 11.9 KB |

| | | | | | | |
|-------------|------|-------------|-----------|-----|------------|---------|
| Testcase 19 | Easy | Hidden case | ✓ Success | 0.4 | 0.0278 sec | 11.9 KB |
| Testcase 20 | Easy | Hidden case | ✓ Success | 0.4 | 0.0303 sec | 12.1 KB |
| Testcase 21 | Easy | Hidden case | ✓ Success | 0.4 | 0.0533 sec | 12.1 KB |
| Testcase 22 | Easy | Hidden case | ✓ Success | 0.4 | 0.0439 sec | 12.3 KB |
| Testcase 23 | Easy | Hidden case | ✓ Success | 0.4 | 0.0293 sec | 12.1 KB |
| Testcase 24 | Easy | Hidden case | ✓ Success | 0.8 | 0.0745 sec | 12.2 KB |

No Comments

QUESTION 3



Correct Answer

Score 10

Know your shopping cart! > Coding CS 101

QUESTION DESCRIPTION

Market basket analysis is a technique in datamining that analyses combinations of products in customers' shopping basket to know which products have been bought together. This sort of analysis has wide applications in the retail industry including, shelf placement, cross-marketing, catalogue design, sale campaign analysis etc.

Intrigued by the idea, you decided to keep track of your daily purchases at home and write a python program to gain some insight of this data. Here is a snapshot of your daily purchases:

| | |
|---|---------------------------------------------|
| 1 | Milk, Tea, Sugar, Tissue Box, Air Freshener |
| 2 | Tea, Bread, Eggs, Butter |
| 3 | Pizza, Pepsi, Chips |
| 4 | Meat, Yogurt, Onion, Tomato, Potatoe |
| 5 | Bun, Patties, Chips, Pepsi |
| 6 | Samosa, Roll, Jalebi |

The goal is to find a list of frequent items for a given item.

Write a function `get_frequent_items(..)` that takes a list of daily transactions (`purchases`) and a given item name (`item`) and returns item(s) that has been bought with `item` most frequently. The function will return a *list* of frequent items.

Input

The function will take two parameters:

- `purchases`: which is a nested list of item names showing daily purchases.
- `item`: an item name

Output

The function will return a list of items most frequent sold with the given item. if there are no such items, an empty list will be returned.

Examples

Input:

```
Pepsi
['Milk', 'Tea', 'Sugar'], ['Tissue Box', 'Air Freshener'], ['Tea', 'Bread', 'Eggs', 'Butter'], ['Pizza', 'Pepsi',
```



[Picture
Credit: [Data Mining Examples](#)]

'Chips'], ['Meat', 'Yogurt', 'Onion', 'Tomato', 'Potatoe'], ['Bun', 'Patties', 'Chips', 'Pepsi'], ['Samosa', 'Roll', 'Jalebi']

Output:

['Chips']

Note:

'Chips' is sold most frequently (two times) with Pepsi and hence is returned as the most frequent item.

Input:

Meat

[['Meat', 'Yogurt', 'Tomato', 'Rice'], ['Meat', 'Onion', 'Yogurt'], ['Yogurt', 'Flour', 'Onion'], ['Sugar', 'Oil', 'Soap', 'Detergent'], ['Eggs', 'Bread', 'Butter', 'Yogurt'], ['Onion', 'Meat', 'Yogurt', 'Spices'], ['Meat', 'Onion', 'Tomato']

Output:

['Onion', 'Yogurt']

Note:

Both 'Onion' and 'Yogurt' were purchased with 'Meat' three times whereas other items are purchased fewer times. Therefore, a list of these two items will be returned.

INTERVIEWER GUIDELINES

```
def get_frequent_items(purchases,item):
    soldtogether = {}
    for lst in purchases:
        if item in lst:
            for i in lst:
                if i !=item:
                    soldtogether[i] = soldtogether.get(i,0) + 1

    if len(soldtogether) == 0:
        return []

    itemslist = []
    maxQty = max(soldtogether.values())

    for i in soldtogether:
        if soldtogether[i] == maxQty:
            itemslist.append(i)
    return itemslist
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2 def get_frequent_items(purchases,item):
3     soldtogether = {}
4     for lst in purchases:
5         if item in lst:
6             for i in lst:
7                 if i !=item:
8                     soldtogether[i] = soldtogether.get(i,0) + 1
```

```

9
10     if len(soldtogether) == 0:
11         return []
12
13     itemslist = []
14     maxQty = max(soldtogether.values())
15
16     for i in soldtogether:
17         if soldtogether[i] == maxQty:
18             itemslist.append(i)
19     return itemslist
20

```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|------------|------------|-------------|-----------|-------|------------|-------------|
| Testcase 0 | Easy | Sample case | ✔ Success | 1 | 0.0218 sec | 10.3 KB |
| Testcase 1 | Easy | Hidden case | ✔ Success | 1 | 0.0237 sec | 10.2 KB |
| Testcase 2 | Easy | Sample case | ✔ Success | 1 | 0.018 sec | 10.3 KB |
| Testcase 3 | Easy | Hidden case | ✔ Success | 1 | 0.0219 sec | 10.2 KB |
| Testcase 4 | Easy | Sample case | ✔ Success | 1 | 0.0188 sec | 10.2 KB |
| Testcase 5 | Easy | Hidden case | ✔ Success | 1 | 0.0197 sec | 10 KB |
| Testcase 7 | Easy | Hidden case | ✔ Success | 1 | 0.0196 sec | 10.3 KB |
| Testcase 8 | Easy | Hidden case | ✔ Success | 1 | 0.0216 sec | 10.2 KB |
| Testcase 9 | Easy | Hidden case | ✔ Success | 2 | 0.019 sec | 10 KB |

No Comments

QUESTION 4



Correct Answer

Score 10

Merge by Key > Coding

QUESTION DESCRIPTION

Problem

Write a function named `merge_key` that takes two dictionaries `d1` and `d2` as parameters and builds a dictionary that contains every key from `d1` and `d2` with the corresponding value. If a key appears in both `d1` and `d2`, the value in the merged dictionary is a list containing the value from `d1` and from `d2`. The function *returns* a sorted list of the (key, value) pairs in the merged dictionary.

Hint: Use Python Dictionary [get](#) method.

Sample

```

>>> d1 = {i:chr(96+i) for i in range(1,11)}
>>> d2 = {i:chr(64+i) for i in range(1,11)}
>>> d1
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g', 8: 'h', 9: 'i',
10: 'j'}
>>> d2
{1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E', 6: 'F', 7: 'G', 8: 'H', 9: 'I',
10: 'J'}
>>> merge_key(d1,d2)
{1: ['a', 'A'], 2: ['b', 'B'], 3: ['c', 'C'], 4: ['d', 'D'], 5: ['e',
'E'], 6: ['f', 'F'], 7: ['g', 'G'], 8: ['h', 'H'], 9: ['i', 'I'], 10:
['j', 'J']}

```

Input Format

The input contains `d1` and `d2` on separate lines.

Output Format

The output should be a sorted list of the (key, value) pairs in the merged dictionary.

INTERVIEWER GUIDELINES

Solution

```
def merge_key(d1, d2):
    d = {}
    # Iterate over the items (k, v) of d1 and d2. Insert every newly
    encountered
    # k into a new dictionary as a key with [v] as the value. If k is
    # encountered again, store the corresponding value in the new
    dictionary in
    # the previously created list.
    for k,v in list(d1.items()) + list(d2.items()):
        # dict.get() eliminates the need for if-else
        d[k] = d.get(k, []) + [v]
    return d
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2 def merge_key(d1, d2):
3     d = {}
4     # Iterate over the items (k, v) of d1 and d2. Insert every newly
5     encountered
6     # k into a new dictionary as a key with [v] as the value. If k is
7     # encountered again, store the corresponding value in the new dictionary
8     in
9     # the previously created list.
10    for k,v in list(d1.items()) + list(d2.items()):
11        # dict.get() eliminates the need for if-else
12        d[k] = d.get(k, []) + [v]
    return d
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|------------|------------|-------------|-----------|-------|------------|-------------|
| Testcase 0 | Easy | Sample case | ✔ Success | 2 | 0.0166 sec | 9.38 KB |
| Testcase 1 | Easy | Hidden case | ✔ Success | 2 | 0.0155 sec | 9.24 KB |
| Testcase 2 | Easy | Hidden case | ✔ Success | 2 | 0.0199 sec | 9.5 KB |
| Testcase 3 | Easy | Hidden case | ✔ Success | 2 | 0.0252 sec | 9.22 KB |
| Testcase 4 | Easy | Hidden case | ✔ Success | 2 | 0.0291 sec | 9.67 KB |

No Comments

QUESTION 5



Correct Answer

Count Words > Coding

QUESTION DESCRIPTION

Problem

Write a function named `count_words` that uses a dictionary to count the words in its parameter named `s` and of type `str`. It then *prints* the identified words in ascending order along with their frequency in `s` as shown in the sample below. Space, case, and special characters must be ignored when counting.

Sample

```
>>> count_words("Python is a widely used high-level programming language
for general-purpose programming, created by Guido van Rossum and first
released in 1991. An interpreted language, Python has a design philosophy
that emphasizes code readability (notably using whitespace indentation to
delimit code blocks rather than curly brackets or keywords), and a syntax
that allows programmers to express concepts in fewer lines of code than
might be used in languages such as C++ or Java. It provides constructs
that enable clear programming on both small and large scales.")
1991 = 1
a = 3
allows = 1
an = 1
and = 3
as = 1
be = 1
blocks = 1
both = 1
brackets = 1
by = 1
c = 1
clear = 1
code = 3
concepts = 1
constructs = 1
created = 1
curly = 1
delimit = 1
design = 1
emphasizes = 1
enable = 1
express = 1
fewer = 1
first = 1
for = 1
generalpurpose = 1
guido = 1
has = 1
highlevel = 1
in = 3
indentation = 1
interpreted = 1
is = 1
it = 1
java = 1
keywords = 1
language = 2
languages = 1
large = 1
lines = 1
might = 1
notably = 1
of = 1
on = 1
or = 2
philosophy = 1
programmers = 1
programming = 3
provides = 1
python = 2
rather = 1
readability = 1
released = 1
```

```
rossum = 1
scales = 1
small = 1
such = 1
syntax = 1
than = 2
that = 3
to = 2
used = 2
using = 1
van = 1
whitespace = 1
widely = 1
```

INTERVIEWER GUIDELINES

Solution

```
def count_words(s):
    new_s = ''
    for letter in s.lower():
        if ord('a') <= ord(letter) <= ord('z') or letter in '01234567890 ':
            new_s += letter
    d = {}
    for word in new_s.split():
        d[word] = d.get(word, 0) + 1
    for k,v in sorted(d.items()):
        print(k, '=', v)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def count_words(s):
2     new_s = ''
3     for letter in s.lower():
4         if ord('a') <= ord(letter) <= ord('z') or letter in '01234567890 ':
5             new_s += letter
6     d = {}
7     for word in new_s.split():
8         d[word] = d.get(word, 0) + 1
9     for k,v in sorted(d.items()):
10         print(k, '=', v)
11
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|------------|------------|-------------|-----------|-------|------------|-------------|
| Testcase 0 | Easy | Sample case | ✔ Success | 3 | 0.0166 sec | 9.6 KB |
| Testcase 1 | Easy | Sample case | ✔ Success | 3 | 0.0145 sec | 9.37 KB |
| Testcase 2 | Easy | Sample case | ✔ Success | 4 | 0.0156 sec | 9.44 KB |

No Comments

QUESTION 6



Contacts > Coding

CS101

Python 3

Correct Answer

Score 10

Background

Most modern smartphones have some notion of a contact list, which allows the user to store one or more phone numbers for a given contact. Since a contact is a unique individual, contact dictionary might be a more appropriate term. You are to implement a rudimentary contact list in Python.

Task

You will be provided a contact name and phone number on each line of input. Write a program that reads each line of input, and generates a contact dictionary accordingly. Each contact name will be stored as key in "last name, first name(s)" format, and phone numbers will be stored as corresponding value as a list. Duplicate numbers will be ignored. Phone list will be kept sorted. See examples below.

Function Description

To implement the given task, write the following function:

1. ***contacts*** that takes no argument, and reads input from standard input stream (use the `input()` function). The function should return a dictionary of names, with corresponding value of a sorted list containing phone numbers. Name must be changed to "last name, first name(s)" format.

Constraints

- Output will be handled by HackerRank--you should not display values yourself.
- Each line of input will contain the contact's full name, a space, and their phone number. Note that the phone number is a string.

Notes

- Input ends with the string STOP by itself on a line.
- Use the `strip` method to remove whitespace from the input.
- For testing purposes, the output will be pretty printed.

▼ Input Format For Custom Testing

There are multiple lines of input--each line except for last contains a string consisting of a full name and one phone number.

The last line of input contains the word STOP by itself.

▼ Sample Case 0

Sample Input For Custom Testing

```
Sona 0300-555-1111
Chandi 0300-555-2222
Chacha Karmoo 0300-333-7777
Maasi Barkatay 0300-222-9999
Nawab Farasat Ali Khan 0388-888-8888
STOP
```

Sample Output

```
{'Barkatay, Maasi': ['0300-222-9999'],
 'Chandi': ['0300-555-2222'],
 'Karmoo, Chacha': ['0300-333-7777'],
 'Khan, Nawab Farasat Ali': ['0388-888-8888'],
 'Sona': ['0300-555-1111']}
```

Explanation

Single names, such as "Sona" and "Chandi", are left as is. Other names with last name and first names are formatted, such as "Barkatay, Maasi".

Each contact has a single number, which is stored in a singleton list.

STOP by itself on a line signals the end of input.

▼ Sample Case 1

Sample Input For Custom Testing

```
Chacha Karmoo 0300-333-7778
Chandi 0300-555-2223
Nawab Farasat Ali Khan 0388-888-8898
Nawab Farasat Ali Khan 0388-888-8899
Chandi 0300-555-2224
Sona 0300-555-1111
Chacha Karmoo 0300-333-7778
Chandi 0300-555-2224
Nawab Farasat Ali Khan 0388-888-8888
Chandi 0300-555-2222
Nawab Farasat Ali Khan 0388-889-8889
Sona 0300-555-1112
Chacha Karmoo 0300-333-7777
Nawab Farasat Ali Khan 0388-888-8888
Nawab Farasat Ali Khan 0388-889-8888
Chandi 0300-555-2222
Chandi 0300-555-2222
Nawab Farasat Ali Khan 0388-888-8889
Nawab Farasat Ali Khan 0388-888-8888
Chacha Karmoo 0300-333-7778
Maasi Barkatay 0300-222-9999
Maasi Barkatay 0300-222-9999
STOP
```

Sample Output

```
{'Barkatay, Maasi': ['0300-222-9999'],
 'Chandi': ['0300-555-2222', '0300-555-2223', '0300-555-2224'],
 'Karmoo, Chacha': ['0300-333-7777', '0300-333-7778'],
 'Khan, Nawab Farasat Ali': ['0388-888-8888',
                             '0388-888-8889',
                             '0388-888-8898',
                             '0388-888-8899',
                             '0388-889-8888',
                             '0388-889-8889'],
 'Sona': ['0300-555-1111', '0300-555-1112']}
```

Explanation

The phone numbers for each contact are stored in sorted order.

Any duplicate contact information is ignored.

INTERVIEWER GUIDELINES

```
def contacts():
    cont = {}
    while True:
        line = input().strip()
        if line == 'STOP':
            return cont
        line = line.split()
        phone = line.pop()
        last_name = line.pop()
        rest_name = ' '.join(line)
        name = last_name + ', ' + rest_name if line else last_name
        if name in cont:
            phones = cont[name]
            if phone not in phones:
                phones.append(phone)
            phones.sort()
        else:
            cont[name] = [phone]
```

CANDIDATE ANSWER

Language used: Python 3

```
1 def contacts():
2     cont = {}
3     while True:
4         line = input().strip()
5         if line == 'STOP':
6             return cont
7         line = line.split()
8         phone = line.pop()
9         last_name = line.pop()
10        rest_name = ' '.join(line)
11        name = last_name + ', ' + rest_name if line else last_name
12        if name in cont:
13            phones = cont[name]
14            if phone not in phones:
15                phones.append(phone)
16            phones.sort()
17        else:
18            cont[name] = [phone]
19
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|------------|------------|-------------|-----------|-------|------------|-------------|
| TestCase 0 | Easy | Sample case | ✔ Success | 1 | 0.0368 sec | 11.5 KB |
| TestCase 1 | Easy | Sample case | ✔ Success | 1 | 0.0263 sec | 11.4 KB |
| TestCase 2 | Easy | Sample case | ✔ Success | 1 | 0.0486 sec | 11.5 KB |
| TestCase 3 | Easy | Sample case | ✔ Success | 1 | 0.0329 sec | 11.4 KB |
| TestCase 4 | Easy | Sample case | ✔ Success | 1 | 0.0265 sec | 11.5 KB |
| TestCase 5 | Easy | Hidden case | ✔ Success | 1 | 0.0299 sec | 11.6 KB |
| TestCase 6 | Easy | Hidden case | ✔ Success | 1 | 0.0671 sec | 11.6 KB |
| TestCase 7 | Easy | Hidden case | ✔ Success | 1 | 0.0628 sec | 11.5 KB |
| TestCase 8 | Easy | Hidden case | ✔ Success | 1 | 0.0924 sec | 11.4 KB |
| TestCase 9 | Easy | Hidden case | ✔ Success | 1 | 0.0273 sec | 11.5 KB |

No Comments

PDF generated at: 17 Nov 2023 20:44:27 UTC