alphabetically smaller, we print it. Use dictionary to solve the above problem

Sample Input:

10

John

John

Johny

Jamie

Jamie

Johny

Jack

Johny

Johny

Jackie

Sample Output:

Johny

Answer: (penalty regime: 0 %)

```
votes = [input().strip() for _ in range(int(input()))]
vote_count = {}
```

	Input	Expected	Got	
✓	10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny	Johny	✓
✓	6 Ida Ida Ida Kiruba Kiruba Kiruba	Ida	Ida	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

For example:

Input	Result

`2`
`Gfg 6 7 4`
`Best 7 6 5`
	`Gfg 17`
	`Best 18`

Answer: (penalty regime: 0 %)

```
test_dict = {}
for _ in range(int(input())):
    entry = input().split()
    key = entry[0]
    values = list(map(int, entry[1:]))
    test_dict[key] = sum(values)
sorted_dict = dict(sorted(test_dict.items(), key=lambda item: item[1]
for k, v in sorted_dict.items():
    print(k, v)
```

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use dictionary to solve the problem

For example:

Input	Result
this apple is sweet this apple is sour	sweet sour

Answer: (penalty regime: 0 %)

```
s1 = input()
s2 = input()
word_count = {}
for word in s1.split() + s2.split():
    word_count[word] = word_count.get(word, 0) + 1
uncommon_words = [word for word in word_count if word_count[word] == 
print(' '.join(uncommon_words))
```

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

For example:

Input	Result
4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith

Answer: (penalty regime: 0 %)

```
def compute_student_statistics(n, student_data):
    student = {}
    for data in student_data:
        parts = data.split()
        name, marks = parts[0], list(map(int, parts[1:]))
        student[name] = marks + [sum(marks) / 3]
        highest_avg, highest_assign, lowest_lab, lowest_avg = [], []
        highest_avg_score = highest_assign_score = float('-inf')
        lowest_lab_score = lowest_avg_score = float('inf')
        for name, marks in student.items():
            avg_score = marks[3]
            if avg_score > highest_avg_score:
```

```
compute_student_statistics(n, student_data)
```

	Input	Expected	Got	
✓	4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith	Ram James Ram Lalith Lalith	✓
✓	3 Raja 95 67 90 Aarav 89 90 90 Shadhana 95 95 91	Shadhana Shadhana Aarav Raja Raja	Shadhana Shadhana Aarav Raja Raja	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

10 Q and Z

Write a program that computes and displays the Scrabble™ score for a word. Create a dictionary that maps from letters to point values. Then use the dictionary to compute the score.

A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

Sample Input

REC

Sample Output

REC is worth 5 points.

For example:

Input	Result
REC	REC is worth 5 points.

Answer: (penalty regime: 0 %)

```
letter_points = {'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R'
                 'D': 2, 'G': 2,
                 'B': 3, 'C': 3, 'M': 3, 'P': 3,
                 'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
                 'K': 5,
                 'J': 8, 'X': 8,
                 'Q': 10, 'Z': 10}

word = input().upper()
score = sum(letter_points[letter] for letter in word)
print(f"{word} is worth {score} points.")
```