09 – Dictionary

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

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

Uncommon words

A sentence is a string of single-space separated words where each word consists only of lowercase letters.A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

```python
def uncommon_words(s1, s2):
    # Split the sentences into words
    words1 = s1.split()
    words2 = s2.split()

    # Create a dictionary to count occurrences of each word
    word_count = {}

    # Count words in the first sentence
    for word in words1:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1

    # Count words in the second sentence
    for word in words2:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1

    # Find words that appear exactly once
    uncommon = [word for word in word_count if word_count[word] == 1]

    for i in uncommon:
        print(i,end=" ")

# Example usage
s1 =input()
s2 =input()
uncommon_words(s1, s2)
```

Input : test_dict = {'Gfg' : [6, 7, 4], 'best' : [7, 6, 5]}

Output : {'Gfg': 17, 'best': 18}

Explanation : Sorted by sum, and replaced.

 Input : test_dict = {'Gfg' : [8,8], 'best' : [5,5]}

Output : {'best': 10, 'Gfg': 16}

Explanation : Sorted by sum, and replaced.

 Sample Input:

2

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

Sort Dictionary by Values Summation

Give a dictionary with value lists, sort the keys by summation of values in value list.

```python
n =int(input())
d = {}

for _ in range(n):
    s = input()
    st = s.split()
    key = st[0]
    values = [int(val) for val in st[1:]]
    d[key] = sum(values)

# Sort the dictionary by values
sorted_d = dict(sorted(d.items(), key=lambda item: item[1]))

# Print the sorted dictionary
for key, value in sorted_d.items():
    print(key, value)
```

Examples:

Input : votes[] = {"john", "johnny", "jackie",
           "johnny", "john", "jackie",
           "jamie", "jamie", "john",
           "johnny", "jamie", "johnny",
           "john"};

Output : John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johny get maximum votes. Since John is 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

For example:

Input	Result
10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny

Winner of Election

Given an array of names of candidates in an election. A candidate name in the array represents a vote cast to the candidate. Print the name of candidates received Max vote. If there is tie, print a lexicographically smaller name.

```python
def find_winner(votes):
    # Create a dictionary to store the count of votes for each candidate
    vote_count = {}

    for vote in votes:
        if vote in vote_count:
            vote_count[vote] += 1
        else:
            vote_count[vote] = 1

    # Find the candidate with the maximum votes
    max_votes = 0
    winner = ""

    for candidate, count in vote_count.items():
        if count > max_votes:
            max_votes = count
            winner = candidate
        elif count == max_votes:
            if candidate < winner:
                winner = candidate

    return winner

# Sample Input
n = int(input())
votes = []
for _ in range(n):
    votes.append(input().strip())

# Find and print the winner
winner = find_winner(votes)
print(winner)
```

Sample input:

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

Student Record

Create a student dictionary for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.
1.Identify the student with the highest average score
2.Identify the student who as the highest Assignment marks
3.Identify the student with the Lowest lab marks
4.Identify the student with the lowest average score
Note:
If more than one student has the same score display all the student names

```
print("\n",end="")
for i,j in d.items():
    if (m==j[1]):
        print(i,end=" ")
print("\n",end="")
l=[]
for i,j in d.items():
    if (n==j[2]):
        l.append(i)
l.sort()
for i in l:
    print(i,end=" ")
print("\n",end="")

for i,j in d.items():
    if (o==(sum(j)/3)):
        print(i,end=" ")
print("\n",end="")print("\n",end="")
for i,j in d.items():
    if (m==j[1]):
        print(i,end=" ")
print("\n",end="")
l=[]
for i,j in d.items():
    if (n==j[2]):
        l.append(i)
l.sort()
for i in l:
    print(i,end=" ")
print("\n",end="")

for i,j in d.items():
    if (o==(sum(j)/3)):
        print(i,end=" ")
print("\n",end="")
```

The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Sample Input

REC

Sample Output

REC is worth 5 points.

Scramble Score

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points.

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.

```
d = {
    'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T': 1, 'U': 1,
    '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
}
s=input()
c=0
for i in s:
    c+=d[i]
print(s,"is worth",c,"points.")
```