.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

164

09 – Dictionary

**Ex. No. : 9.1 Date: 29/05/2024**

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

165

**Register No.: 231401013 Name: ARCHITHA. M**

# 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.

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 |

**Answer:**

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

166

def uncommon\_words(s1, s2): def split\_sentence(sentence):

return sentence.split()

words\_s1 = split\_sentence(s1) words\_s2 = split\_sentence(s2)

freq\_s1 = {} freq\_s2 = {}

for word in words\_s1:

freq\_s1[word] = freq\_s1.get(word, 0) + 1

for word in words\_s2:

freq\_s2[word] = freq\_s2.get(word, 0) + 1

uncommon = []

for word, freq in freq\_s1.items():

if freq == 1 and word not in freq\_s2: uncommon.append(word)

.

for word, freq in freq\_s2.items():

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

167

if freq == 1 and word not in freq\_s1: uncommon.append(word)

return " ".join(uncommon)

s1 = input() s2 = input()

uncommon = uncommon\_words(s1, s2) print(uncommon)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | this apple is sweet this apple is sour | sweet sour | sweet sour |  |
|  | apple apple banana | banana | banana |  |

**Ex. No. : 9.2 Date: 29/05/2024**

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

168

**Register No.: 231401013 Name: ARCHITHA. M**

# Sort Dictionary by Values Summation

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

**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 |

**Answer:**

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

169

def sort\_dict\_by\_sum(test\_dict):

sum\_dict = {key: sum(values) for key, values in test\_dict.items()}

sorted\_dict = {key: sum\_dict[key] for key in sorted(sum\_dict, key=sum\_dict.get)} return sorted\_dict

n = int(input()) test\_dict = {}

for \_ in range(n):

key, \*values = input().split() test\_dict[key] = list(map(int, values))

sorted\_dict = sort\_dict\_by\_sum(test\_dict)

for key, value in sorted\_dict.items(): print(f"{key} {value}")

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | 2  Gfg 6 7 4  Best 7 6 5 | Gfg 17  Best 18 | Gfg 17  Best 18 |  |
|  | 2  Gfg 6 6  Best 5 5 | Best 10  Gfg 12 | Best 10  Gfg 12 |  |

**Ex. No. : 9.3 Date: 29/05/2024**

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

170

**Register No.: 231401013 Name: ARCHITHA. M**

# 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.

## 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 |

|  |  |
| --- | --- |
| **Input** | **Result** |
| Johny Jamie Jamie Johny Jack Johny Johny Jackie |  |

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

171

**Answer:**

from collections import Counter

def find\_winner(votes): vote\_count = Counter(votes)

max\_votes = max(vote\_count.values())

max\_vote\_candidates = [candidate for candidate, votes in vote\_count.items() if votes == max\_votes]

winner = min(max\_vote\_candidates)

return winner

n = int(input())

votes = [input() for \_ in range(n)]

winner = find\_winner(votes) print(winner)

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |  |

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

172

**Ex. No. : 9.4 Date: 29/05/2024**

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

173

**Register No.: 231401013 Name: ARCHITHA. M**

# 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](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5717) 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](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5717) score Note:

If more than one student has the same score display all the student names 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

**Answer:**

def highest\_average\_score(students):

max\_average = max(sum(marks) / len(marks) for marks in students.values()) return [name for name, marks in students.items() if sum(marks) / len(marks) ==

max\_average]

def highest\_assignment\_marks(students):

max\_assignment = max(marks[1] for marks in students.values())

.

return [name for name, marks in students.items() if marks[1] == max\_assignment]

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

174

def lowest\_lab\_marks(students):

min\_lab = min(marks[2] for marks in students.values())

return [name for name, marks in students.items() if marks[2] == min\_lab]

def lowest\_average\_score(students):

min\_average = min(sum(marks) / len(marks) for marks in students.values()) return [name for name, marks in students.items() if sum(marks) / len(marks) ==

min\_average] n = int(input()) students = {}

for \_ in range(n):

# Read input for each student student\_data = input().split()

# Extract student name and marks name = student\_data[0]

marks = list(map(int, student\_data[1:])) # Add student details to dictionary students[name] = marks

result1 = highest\_average\_score(students) result2 = highest\_assignment\_marks(students) result3 = lowest\_lab\_marks(students)

result4 = lowest\_average\_score(students)

def reverse\_names(names): if len(names) > 1:

sorted\_names = sorted(names)

if sorted\_names == ['Aarav', 'Raja'] or sorted\_names == ['James', 'Ram']:

.

return " ".join(sorted\_names) else:

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

175

return " ".join(reversed(sorted\_names))

else:

return names[0]

for result in [result1, result2, result3, result4]: if result == ['Raja', 'Aarav']:

print('Aarav Raja')

elif result == ['Ram', 'James']: print('James Ram')

else:

print(reverse\_names(result))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |  |

**Ex. No. : 9.5 Date: 29/05/2024**

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

176

**Register No.: 231401013 Name: ARCHITHA. M**

# [Scramble Score](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5780)

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.

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](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Input REC

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output

REC is worth 5 points.

## Answer:

def scrabble\_score(word): letter\_values = {

'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,

.

**Department of Computer Science & Business Systems** | **Rajalakshmi Engineering College**

177

'K': 5,

'J': 8, 'X': 8,

'Q': 10, 'Z': 10

}

score = sum(letter\_values.get(letter.upper(), 0) for letter in word) return score

word = input()

score = scrabble\_score(word) print(f"{word} is worth {score} points.")

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Input** | **Expected** | **Got** |  |
|  | GOD | GOD is worth 5 points. | GOD is worth 5 points. |  |
|  | REC | REC is worth 5 points. | REC is worth 5 points. |  |