# 09 – Dictionary

**Ex. No. : 9.1** **Date: 01.06.24**

**Register No.: 231901020** **Name: KAVIYA.V**

---

## Uncommon words

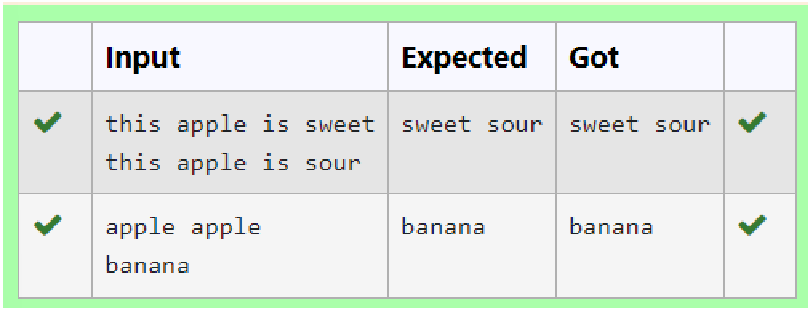
A sentence is a string of single-space separated words where each word consists only of lowercase letters.A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

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

### CODING:

```
s1=input().strip()

s2=input().strip()

ws1=s1.split()

ws2=s2.split()

wcs1={}

wcs2={}

for i in ws1:

    wcs1[i]=wcs1.get(i,0)+1


for i in ws2:

    wcs2[i]=wcs2.get(i,0)+1

uncommon_words=set()
```

```
for w,c in wcs1.items():
    if c==1 and w not in wcs2:
        uncommon_words.add(w)
for w,c in wcs2.items():
    if c==1 and w not in wcs1:
        uncommon_words.add(w)
if len(uncommon_words)==0:
    print("No uncommon words")
else:
    print(*uncommon_words)
```



| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | this apple is sweet<br>this apple is sour | sweet sour | sweet sour | ✓ |
| ✓ | apple apple<br>banana | banana | banana | ✓ |

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<br>this apple is sour | sweet sour |

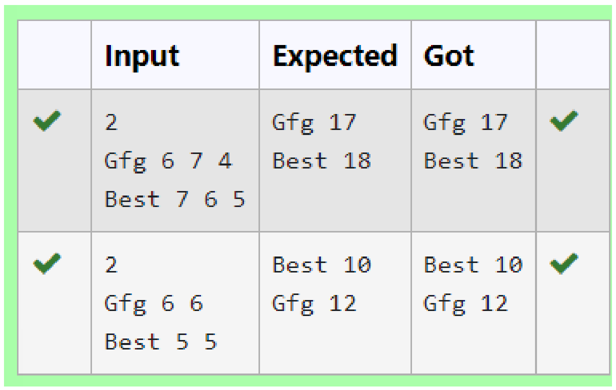
| | | | |
|---|---|---|---|
| **Ex. No. :** | **9.2** | **Date:** | **01.06.24** |
| **Register No.:** | **231901020** | **Name:** | **KAVIYA.V** |

## Sort Dictionary by Values Summation

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

**CODING:**

```
n=int(input())
d={}
for _ in range(n):
   data=input().split()
   key=data[0]
   values=list(map(int,data[1:]))
   d[key]=values
sum_d={}
for key,values in d.items():
   sum_d[key]=sum(values)
sort=dict(sorted(sum_d.items(),key=lambda
item:item[1]))
for key,total in sort.items():
   print(f"{key} {total}")
```



| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 2<br>Gfg 6 7 4<br>Best 7 6 5 | Gfg 17<br>Best 18 | Gfg 17<br>Best 18 | ✓ |
| ✓ | 2<br>Gfg 6 6<br>Best 5 5 | Best 10<br>Gfg 12 | Best 10<br>Gfg 12 | ✓ |

**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<br>Gfg 6 7 4<br>Best 7 6 5 | Gfg 17<br>Best 18 |

**Ex. No. : 9.3** **Date: 01.06.24**

**Register No.: 231901020** **Name: KAVIYA.V**

---

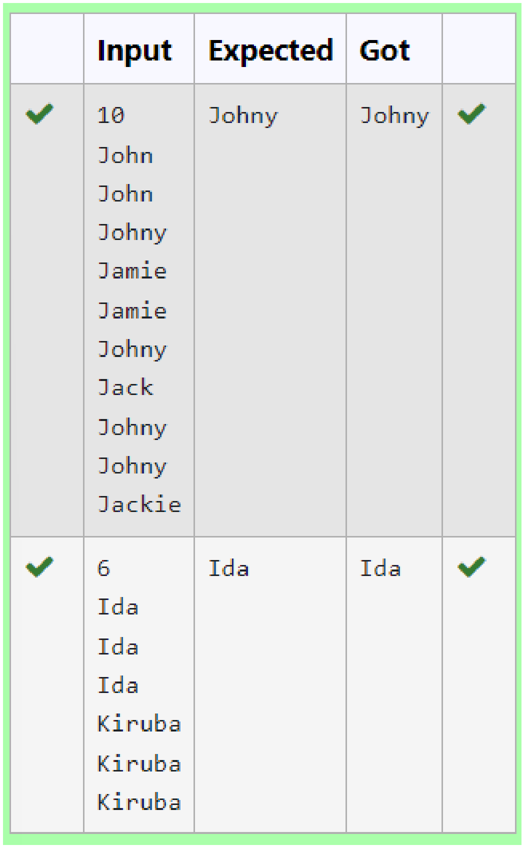
## Winner of Election

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.

## CODING:

```
try:
    n=int(input())
    vd={}
    for i in range(n):
        c=input()
        if c in vd:
            vd[c]+=1
        else:
            vd[c]=1
    mv=max(vd.values())
    w=[c for c,v in vd.items() if v==mv]
    w=min(w)
    print(w)
```

```
except EOFError:
    print("No input provided.")
```



| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 10<br>John<br>John<br>Johny<br>Jamie<br>Jamie<br>Johny<br>Jack<br>Johny<br>Johny<br>Jackie | Johny | Johny | ✓ |
| ✓ | 6<br>Ida<br>Ida<br>Ida<br>Kiruba<br>Kiruba<br>Kiruba | Ida | Ida | ✓ |

**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<br>John<br>John<br>Johny<br>Jamie<br>Jamie<br>Johny<br>Jack<br>Johny<br>Johny<br>Jackie | Johny |

**Ex. No. : 9.4** **Date: 01.06.24**

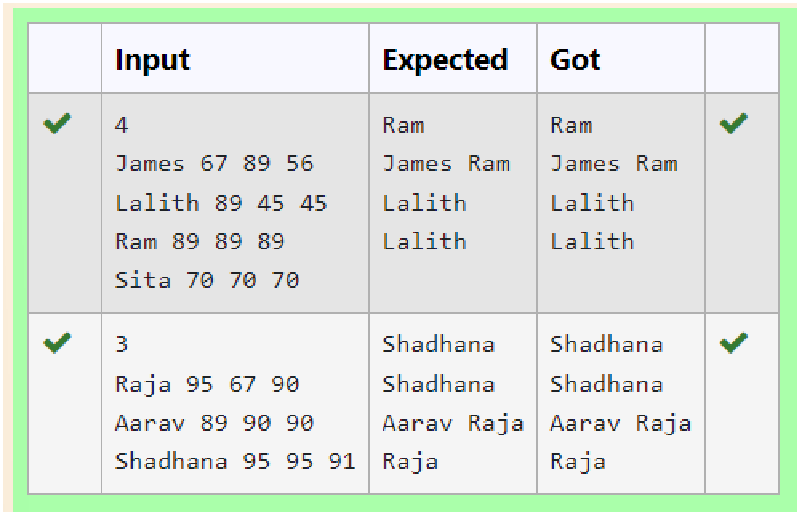
**Register No.: 231901020** **Name: KAVIYA.V**

## 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

### CODING:

```
n = int(input())

highest_avg_score = -1

highest_avg_students = []

highest_assignment_score = -1

highest_assignment_students = []

lowest_lab_score = float('inf')

lowest_lab_students = []

lowest_avg_score = float('inf')

lowest_avg_students = []

for _ in range(n):
    name, test_mark, assignment_mark, lab_mark =
input().split()
    test_mark = int(test_mark)
    assignment_mark = int(assignment_mark)
    lab_mark = int(lab_mark)
    avg_score = (test_mark + assignment_mark +
lab_mark) / 3
    if avg_score > highest_avg_score:
        highest_avg_score = avg_score
        highest_avg_students = [name]
    elif avg_score == highest_avg_score:
        highest_avg_students.append(name)
    if assignment_mark > highest_assignment_score:
        highest_assignment_score = assignment_mark
        highest_assignment_students = [name]
    elif assignment_mark == highest_assignment_score:
        highest_assignment_students.append(name)
    if lab_mark < lowest_lab_score:
        lowest_lab_score = lab_mark
        lowest_lab_students = [name]
    elif lab_mark == lowest_lab_score:
        lowest_lab_students.append(name)
    if avg_score < lowest_avg_score:
        lowest_avg_score = avg_score
        lowest_avg_students = [name]
    elif avg_score == lowest_avg_score:
        lowest_avg_students.append(name)
print(' '.join(highest_avg_students))
print(' '.join(highest_assignment_students))
print(' '.join(sorted(lowest_lab_students)))
print(' '.join(lowest_avg_students))
```



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

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
```

**Ex. No. : 9.5** **Date: 01.06.24**

**Register No.: 231901020** **Name: KAVIYA.V**

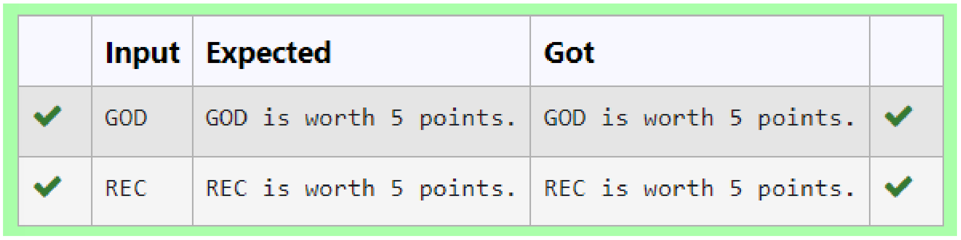
---

## 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.

### CODING:

```
scrabble={'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().upper()
points=0
for i in s:
    if i in scrabble:
        points+=scrabble[i]
    else:
        print("Letter not found.....")
print("%s is worth %d points."%(s,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. | ✓ |

## 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.