**Week8:DICTIONARY AND ITS OPERATIONS**

1. 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}

**Input :** test\_dict = {‘Gfg’ : [8,8], ‘best’ : [5,5]}

**Output :** {‘best’: 10, ‘Gfg’: 16}

**Sample Input:**

2

Gfg 6 7 4

Best 7 6 5

**Sample Output**

Gfg 17

Best 18

**PROGRAM:**

try:

T=int(input())

result\_dict={}

for \_ in range(T):

key,\*values=input().split()

values=list(map(int,values))

sum\_values=sum(values)

result\_dict[key]=sum\_values

sorted\_result=dict(sorted(result\_dict.items(),key=lambda

for key,value in sorted\_result.items():

print(key,value)

except EOFError:

print("No input provided.")

**OUTPUT:**

****

2. 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.

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

**Output:** ["sweet","sour"]

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

**Output:** ["banana"]  
  
**PROGRAM:**

s1=input().strip()

s2=input().strip()

words\_s1=s1.split()

words\_s2=s2.split()

word\_count\_s1={}

word\_count\_s2={}

for word in words\_s1:

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

for word in words\_s2:

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

uncommon\_words=set()

for word, count in word\_count\_s1.items():

if count == 1 and word not in word\_count\_s2:

uncommon\_words.add(word)

for word,count in word\_count\_s2.items():

if count==1 and word not in word\_count\_s1:

uncommon\_words.add(word)

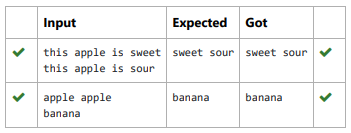
if len(uncommon\_words)==0:

print("No uncommon words")

else:

print(\*uncommon\_words)

**OUTPUT:**

****

3. 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.

**Sample Input:**

10

John

John

Johny

Jamie

Jamie

Johny

Jack

Johny

Johny

Jackie

**Sample Output:**

Johny

**PROGRAM:**

try:

n=int(input())

votes\_dict={}

for \_ in range(n):

candidate=input()

if candidate in votes\_dict:

votes\_dict[candidate]+=1

else:

votes\_dict[candidate]=1

max\_votes=max(votes\_dict.values())

winners=[candidate for candidate,votes in votes\_dict.item

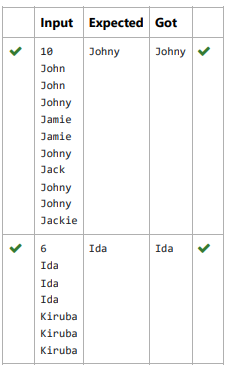
winner=min(winners)

print(winner)

except EOFError:

print("No input provided.")

**OUTPUT:**

****

4. 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. 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

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.

**PROGRAM:**

letter\_values={

'A':1,'E':1,'I':1,'L':1,'N':1,'O':1,'R':1,'S':1,'T':1,'U

'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

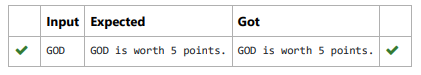
}

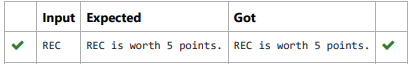
word=input()

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

print(f"{word} is worth {score} points.")

**OUTPUT:**





5. 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

**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

**PROGRAM:**

n=int(input())

student\_data={}

for i 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)

student\_data[name]=(test\_mark,assignment\_mark,lab\_mark

highest\_avg\_score=max(sum(marks)/3 for marks in student\_dat

highest\_avg\_score\_students=[name for name,marks in student\_

highest\_assignment\_marks=max(student\_data.values(),key=lamb

highest\_assignment\_marks\_students=[name for name, marks in

lowest\_lab\_marks=min(student\_data.values(),key=lambda x:x[2

lowest\_lab\_marks\_students=[name for name,marks in student\_d

lowest\_avg\_score=min(sum(marks)/3 for marks in student\_data

lowest\_avg\_score\_students=[name for name,marks in student\_d

highest\_avg\_score\_students.sort()

highest assignment marks students.sort()

lowest\_lab\_marks\_students.sort()

lowest\_avg\_score\_students.sort()

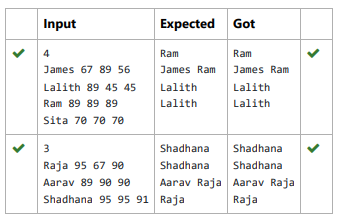
print(\*highest\_avg\_score\_students)

print(\*highest\_assignment\_marks\_students)

print(\*lowest\_lab\_marks\_students)

print(\*lowest\_avg\_score\_students)

**OUTPUT:**

****