09 – Dictionary

# Ex. No. : 9.1 Date:

# Register No.: Name:

# 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 sweetthis apple is sour | sweet sour |

# PROGRAM:

# def uncommon\_words(s1, s2):

# words\_s1 = s1.split()

# words\_s2 = s2.split()

# word\_freq = {}

# for word in words\_s1:

# if word in word\_freq:

# word\_freq[word] += 1

# else:

# word\_freq[word] = 1

# for word in words\_s2:

# if word in word\_freq:

# word\_freq[word] += 1

# else:

# word\_freq[word] = 1

# uncommon\_words = []

# for word, freq in word\_freq.items():

# if freq == 1:

# uncommon\_words.append(word)

# return uncommon\_words

# s1 = input()

# s2 = input()

# print(\*uncommon\_words(s1, s2))

# 

# Ex. No. : 9.2 Date:

# Register No.: Name:

# 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 |

# PROGRAM:

# n = int(input())

# test\_dict = {}

# for \_ in range(n):

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

# values = list(map(int, values))

# test\_dict[key] = sum(values)

# sorted\_dict = dict(sorted(test\_dict.items(), key=lambda item: item[1]))

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

# print(key, value)

# 

# Ex. No. : 9.3 Date:

# Register No.: Name:

**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 |
| --- | --- |
| 10JohnJohnJohnyJamieJamieJohnyJackJohnyJohnyJackie | Johny |

# PROGRAM:

# def max\_vote\_candidate(votes):

# count = {}

# 

# 

# for name in votes:

# count[name] = count.get(name, 0) + 1

# 

# 

# max\_votes = max(count.values())

# 

# 

# max\_candidates = [name for name, votes in count.items() if votes == max\_votes]

# 

# 

# return min(max\_candidates)

# if \_\_name\_\_ == "\_\_main\_\_":

# 

# n = int(input())

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

# 

# 

# print(max\_vote\_candidate(votes))

# 

# Ex. No. : 9.4 Date:

# Register No.: Name:

# 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

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 |

# PROGRAM:

# n = int(input())

# students = {}

# for \_ in range(n):

# name, test\_mark, assignment\_mark, lab\_mark = input().split()

# test\_mark, assignment\_mark, lab\_mark = int(test\_mark), int(assignment\_mark), int(lab\_mark)

# students[name] = {'test\_mark': test\_mark, 'assignment\_mark': assignment\_mark, 'lab\_mark': lab\_mark}

# avg\_scores = {name: sum(details.values()) / 3 for name, details in students.items()}

# highest\_avg\_score = max(avg\_scores, key=avg\_scores.get)

# lowest\_avg\_score = min(avg\_scores, key=avg\_scores.get)

# highest\_assignment\_score = max(students, key=lambda x: students[x]['assignment\_mark'])

# lowest\_lab\_score = min(students, key=lambda x: students[x]['lab\_mark'])

# highest\_avg\_students = [name for name, avg in avg\_scores.items() if avg == avg\_scores[highest\_avg\_score]]

# lowest\_avg\_students = [name for name, avg in avg\_scores.items() if avg == avg\_scores[lowest\_avg\_score]]

# highest\_assignment\_score = max(students, key=lambda x: students[x]['assignment\_mark'])

# highest\_assignment\_students = sorted([name for name, details in students.items() if details['assignment\_mark'] == students[highest\_assignment\_score]['assignment\_mark']])

# lowest\_lab\_score = min(students, key=lambda x: students[x]['lab\_mark'])

# lowest\_lab\_students = sorted([name for name, details in students.items() if details['lab\_mark'] == students[lowest\_lab\_score]['lab\_mark']])

# print(\*highest\_avg\_students)

# print(\*highest\_assignment\_students)

# print(\*lowest\_lab\_students)

# print(\*lowest\_avg\_students)

# 

# Ex. No. : 9.5 Date:

# Register No.: Name:

# [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.

For example:

| Input | Result |
| --- | --- |
| REC | REC is worth 5 points. |

PROGRAM:

word = input().strip().upper()

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,

'K': 5,

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

'Q': 10, 'Z': 10

}

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

print(word,"is worth",score,"points.")

