Started on	Tuesday, 28 May 2024, 1:33 PM
State	Finished
Completed on	Tuesday, 28 May 2024, 2:32 PM
Time taken	59 mins 4 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```
1
2 from collections import defaultdict
```

```
from collections import defaultdict

def find_winner(votes):
    # Create a dictionary to store the count of each candidate's votes
    count = defaultdict(int)

    # Count the votes for each candidate
    for candidate in votes:
        count[candidate] += 1

    # Find the candidate(s) with the maximum number of votes
    max_votes = max(count.values())
    winners = [candidate for candidate, votes in count.items() if votes == max_votes]

    # If there's a tie, return the lexicographically smallest candidate
    return min(winners)

# Take user input for number of votes
n = int(input())

# Take user input for each vote
votes = []
for _ in range(n):
    vote = input()
    votes.append(vote)

# Output
print(find_winner(votes))
```

	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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

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

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

Answer: (penalty regime: 0 %)

```
def highest_average_score(students):
    averages = {name: sum(data.values()) / len(data) for name, data in students.items()}
    max_avg = max(averages.values())
    return ' '.join(sorted([name for name, avg in averages.items() if avg == max_avg]))

def highest_assignment_marks(students):
    max_assignment = max(student['assignment'] for student in students.values())
    return ' '.join(sorted([name for name, data in students.items() if data['assignment'] == max_assignment]))

def lowest_lab_marks(students):
    min_lab = min(student['lab'] for student in students.values())
    return ' '.join(sorted([name for name, data in students.items() if data['lab'] == min_lab]))
```

```
    return ' '.join(sorted([name for name, data in students.items() if data['lab'] == min_lab]))

def lowest_average_score(students):
    averages = {name: sum(data.values()) / len(data) for name, data in students.items()}
    min_avg = min(averages.values())
    return ' '.join(sorted([name for name, avg in averages.items() if avg == min_avg]))

# Function to get student details from user
def get_student_details(num_students):
    students = {}
    for i in range(num_students):
        details = input().split()
        name = details[0]
        marks = {'test': int(details[1]), 'assignment': int(details[2]), 'lab': int(details[3])}
        students[name] = marks
    return students

# Taking input from the user for the number of students
num_students = int(input())

# Getting details for each student
students = get_student_details(num_students)

# Displaying the results
print(highest_average_score(students))
print(highest_assignment_marks(students))
print(lowest_lab_marks(students))
print(lowest_average_score(students))
```

	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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

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.

For example:

Input	Result
REC	REC is worth 5 points.

Answer: (penalty regime: 0 %)

```
def scrabble_score(word):
    # Dictionary mapping letters to their point values
    letter_points = {
        '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
    }

    # Calculate the score for the word
    score = sum(letter_points.get(letter.upper(), 0) for letter in word)

    return score

# Input word
word = input()

# Calculate and display the Scrabble score
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.	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

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: (penalty regime: 0 %)

```
def uncommon_words(s1, s2):
    def count_words(sentence):
        word_count = {}
        words = sentence.split()
        for word in words:
            word_count[word] = word_count.get(word, 0) + 1
        return word_count

    s1_words = count_words(s1)
    s2_words = count_words(s2)

    uncommon = []
    for word, count in s1_words.items():
        if count == 1 and word not in s2_words:
            uncommon.append(word)

    for word, count in s2_words.items():
        if count == 1 and word not in s1_words:
            uncommon.append(word)

    return uncommon

# Input from the user
s1 = input()
s2 = input()

```

```
# Get uncommon words
uncommon = uncommon_words(s1, s2)

# Displaying the uncommon words without square brackets and commas
print(" ".join(uncommon))

```

	Input	Expected	Got	
✓	this apple is sweet this apple is sour	sweet sour	sweet sour	✓
✓	apple apple banana	banana	banana	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

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: (penalty regime: 0 %)

```
def sort_dict_by_sum(test_dict):
    # Calculate the sum of values for each key
    sum_dict = {key: sum(value) for key, value in test_dict.items()}

    # Sort the dictionary keys based on the sums
    sorted_keys = sorted(sum_dict, key=sum_dict.get)

    # Reconstruct the dictionary with sorted keys and their corresponding sums
    sorted_dict = {key: sum_dict[key] for key in sorted_keys}

    return sorted_dict

# Get input from the user
n = int(input())
test_dict = {}
for _ in range(n):
    key, *values = input().split()
    values = list(map(int, values))
    test_dict[key] = values

# Sort the dictionary by sum of values
sorted_dict = sort_dict_by_sum(test_dict)

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

```
27      print(key, value)
28
```

	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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week8_MCQ

Jump to...

Functions ►