WEEK-8 PYTHON PROGRAMMING

1.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

SOLUTION:

```
a=input().split()
b=input().split()
list=[]
for i in a:
    list.append(i)
```

```
for i in b:
    list.append(i)
list1=[]
removed=[]
for i in list:
    if i not in list1:
        list1.append(i)
    else:
        removed.append(i)
for i in removed:
    if i in list1:
        list1.remove(i)
        removed.remove(i)
for i in list1:
    if i!='apple':
        print(i,end=" ")
```

2.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

SOLUTION:

```
def process_and_sort_dict(n, input_data):
    # Create dictionary from input data
    test_dict = {}
    for data in input_data:
        parts = data.split()
        key = parts[0]
        values = list(map(int, parts[1:]))
        test_dict[key] = values

    # Create a dictionary with sums of the lists
    sum_dict = {key: sum(values) for key, values in test_dict.items()}

    # Sort the dictionary by the sum of the values
    sorted_sum_dict = dict(sorted(sum_dict.items(), key=lambda item: item[1]))

    return sorted_sum_dict

# Main function to handle the input as depicted in the example
def main():
    n = int(input())
    input_data = [input() for _ in range(n)]

    result = process_and_sort_dict(n, input_data)

    for key, value in result.items():
        print(f"{key} {value}")

# Run the main function
main()
```

3.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.`

SOLUTION:

```
def score(word):
    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
    }
    word = word.upper()
    total_score = sum(letter_points.get(letter) for letter in word)
    return total_score
word = input()
score =score(word)
print(f"{word} is worth {score} points.")
```

4.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

SOLUTION:

```
def find_winner(votes):
    vote_count = {}

    # Count votes for each candidate
    for vote in votes:
        if vote in vote_count:
            vote_count[vote] += 1
        else:
            vote_count[vote] = 1

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

    # Return the lexicographically smallest candidate
    return min(candidates_with_max_votes)

# Function to handle multiple test cases
def process_test_cases():
    import sys
    input = sys.stdin.read
    data = input().strip().split()

    # Number of votes
    n = int(data[0])

    # List of votes
    votes = data[1:n+1]

    # Get the winner
    winner = find_winner(votes)
    print(winner)

# Calling the function to process test cases
process_test_cases()
```

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

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

SOLUTION:

```
def process_student_data(n, student_data):
    students = {}

    # Read student data
    for data in student_data:
        name, test_mark, assignment_mark, lab_mark = data.split()
        test_mark = int(test_mark)
        assignment_mark = int(assignment_mark)
        lab_mark = int(lab_mark)
        students[name] = {
            'test_mark': test_mark,
            'assignment_mark': assignment_mark,
            'lab_mark': lab_mark,
            'average': (test_mark + assignment_mark + lab_mark) / 3
        }

    # Identify the student with the highest average score
    highest_average_students = []
    highest_average = -1
    for name, marks in students.items():
        if marks['average'] > highest_average:
            highest_average = marks['average']
            highest_average_students = [name]
        elif marks['average'] == highest_average:
            highest_average_students.append(name)

    # Identify the student who has the highest Assignment marks
    highest_assignment_students = []
    highest_assignment = -1
    for name, marks in students.items():
        if marks['assignment_mark'] > highest_assignment:
            highest_assignment = marks['assignment_mark']
            highest_assignment_students = [name]
        elif marks['assignment_mark'] == highest_assignment:
  highest_assignment_students.append(name)

    # Identify the student with the lowest lab marks
    lowest_lab_students = []
    lowest_lab = 101
    for name, marks in students.items():
        if marks['lab_mark'] < lowest_lab:
            lowest_lab = marks['lab_mark']
            lowest_lab_students = [name]
```

```
        elif marks['lab_mark'] == lowest_lab:
            lowest_lab_students.append(name)
    lowest_lab_students.sort()  # Sort alphabetically

    # Identify the student with the lowest average score
    lowest_average_students = []
    lowest_average = 101
    for name, marks in students.items():
        if marks['average'] < lowest_average:
            lowest_average = marks['average']
            lowest_average_students = [name]
        elif marks['average'] == lowest_average:
            lowest_average_students.append(name)

    return {
        'highest_average': highest_average_students,
 'highest_assignment': highest_assignment_students,
        'lowest_lab': lowest_lab_students,
        'lowest_average': lowest_average_students
    }

# Main function to handle the input as depicted in the example
def main():
    n = int(input())
    student_data = [input() for _ in range(n)]

    result = process_student_data(n, student_data)

    print(" ".join(result['highest_average']))
    print(" ".join(result['highest_assignment']))
    print(" ".join(result['lowest_lab']))
    print(" ".join(result['lowest_average']))

# Run the main function
main()
```