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.

```
words1 = input().split()
words2 = input().split()

word_count = {}

for word in words1:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1

for word in words2:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1

uncommon_words = []

for word in word_count:
    if word_count[word] == 1:
        if word in words1 and word not in words2:
            uncommon_words.append(word)
        if word in words2 and word not in words1:
            uncommon_words.append(word)

for i in uncommon_words:
    print(i,end=" ")
```

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.

```
n = int(input())
stu={}
li=[]
for i in range(n):
    ab=input().split()
    c=0
    for i in range(1,len(ab)):
        c+=int(ab[i])
    stu[ab[0]]=c
    li.append(c)
li.sort()
name=[]
for i in li:
    s=str([k for k,j in stu.items() if j==i])
    print(s[2:-2],i
```

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.

```
a=int(input())
seq={}
for i in range(a):
    s=input()
    if s in seq:
        seq[s]+=1
    else:
        seq[s]=1
max_votes = 0
winners = []

for i,j in seq.items():
    if j > max_votes:
        max_votes = j
        winners = [i]
    elif j == max_votes:
        winners.append(i)

winner = min(winners)

print(winner)
```

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 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

```
n = int(input())
students = {}
for i in range(n):
    ab=input().split()
    students[ab[0]]=[int(ab[1]),int(ab[2]),int(ab[3])]

highest_avg = -1
highest_avg_students = []

highest_assignment = -1
highest_assignment_students = []

lowest_lab = 101
lowest_lab_students = []

lowest_avg = 101
lowest_avg_students = []

for name, marks in students.items():
    avg_score = sum(marks) / 3
    if avg_score > highest_avg:
        highest_avg = avg_score
        highest_avg_students = [name]
    elif avg_score == highest_avg:
        highest_avg_students.append(name)

    if marks[1] > highest_assignment:
        highest_assignment = marks[1]
        highest_assignment_students = [name]
    elif marks[1] == highest_assignment:
        highest_assignment_students.append(name)

if marks[2] < lowest_lab:
lowest_lab = marks[2]
lowest_lab_students = [name]
elif marks[2] == lowest_lab:
lowest_lab_students.append(name)
if avg_score < lowest_avg:
lowest_avg = avg_score
lowest_avg_students = [name]
elif avg_score == lowest_avg:
lowest_avg_students.append(name)
print(" ".join(sorted(highest_avg_students)))
print(" ".join(sorted(highest_assignment_students)))
print(" ".join(sorted(lowest_lab_students)))
print(" ".join(sorted(lowest_avg_students))
```

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

Register No.: **Name:**

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.

```
a=input()
c=0
for i in a:
    if i in ['A', 'E', 'I', 'L', 'N', 'O', 'R', 'S', 'T','U']:
        c+=1
    elif i in ['D','G']:
        c+=2
    elif i in ['B','C','M','P']:
        c+=3
    elif i in ['F','H','V','W','Y']:
        c+=4
    elif i=="K":
        c+=5
    elif i in ['J','X']:
        c+=8
    elif i in ['Q','Z']:
        c+=10
print(a,"is worth",c,"points.")
```