09 – Dictionary

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

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.

Program:

```
def uncommonFromSentences(s1, s2):
    from collections import Counter

    words_s1 = s1.split()
    words_s2 = s2.split()

    count_s1 = Counter(words_s1)
    count_s2 = Counter(words_s2)

    combined_count = count_s1 + count_s2

    uncommon_words = [word for word in combined_count if
    combined_count[word] == 1]

    return " ".join(uncommon_words)

s1 = input()
s2 = input()

print(uncommonFromSentences(s1, s2))
```

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

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.

Program:

```
n = int(input(""))
test_dict = {}
for _ in range(n):
    key, *values = input().split()
    test_dict[key] = list(map(int, values))

sums = {key: sum(values) for key, values in test_dict.items()}

sorted_dict = dict(sorted(sums.items(), key=lambda item: item[1]))

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

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
10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny

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.

Program:

```
num_votes = int(input(""))

vote_count = {}

for _ in range(num_votes):
    vote = input()
    vote_count[vote] = vote_count.get(vote, 0) + 1

max_votes = max(vote_count.values())

winner = ''
max_votes = 0
for candidate in vote_count:
    if vote_count[candidate] > max_votes:
        max_votes = vote_count[candidate]
        winner = candidate
    elif vote_count[candidate] == max_votes and candidate < winner:
        winner = candidate


print(winner)
```

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

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

Program:

```
def process_student_marks():
    n = int(input(""))

    students = {}

    for _ in range(n):
        data = input().strip()
        name, test_mark, assignment_mark, lab_mark = data.split()
        students[name] = {
            'test_mark': int(test_mark),
            'assignment_mark': int(assignment_mark),
            'lab_mark': int(lab_mark)
        }

    averages = {name: (marks['test_mark'] + marks['assignment_mark'] +
    marks['lab_mark']) / 3
                for name, marks in students.items()}
    max_average = max(averages.values())
    highest_avg_students = sorted([name for name, avg in averages.items() if
avg     == max_average])
    max_assignment = max(students[name]['assignment_mark'] for name in
    students)
```

```
    highest_assignment_students = sorted([name for name in students if
    students[name]['assignment_mark'] == max_assignment])

    min_lab = min(students[name]['lab_mark'] for name in students)
    lowest_lab_students = sorted([name for name in students if students[name]
     ['lab_mark'] == min_lab])

    min_average = min(averages.values())
    lowest_avg_students = sorted([name for name, avg in averages.items() if avg
     == min_average])

    print(" ".join(highest_avg_students))
    print(" ".join(highest_assignment_students))
    print(" ".join(lowest_lab_students))
    print(" ".join(lowest_avg_students))

process_student_marks()
```

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 Input

REC

Sample Output

REC is worth 5 points.

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.

Program:

```
points_groups = {
    1: "AEILNORSTU",
    2: "DG",
    3: "BCMP",
    4: "FHVWY",
    5: "K",
    8: "JX",
    10: "QZ"}
letter_to_points = {}
for points in points_groups:
    letters = points_groups[points]
    for letter in letters:
        letter_to_points[letter] = points
word = input("")
word = word.upper()
score = 0
for letter in word:
    if letter in letter_to_points:
        score += letter_to_points[letter]
    else:
        score += 0
print(f"{word} is worth {score} points.")
```

10 - Searching & Sorting

For example:

Input	Result
5 6 5 4 3 8	3 4 5 6 8

Ex. No. : 10.1 **Date:**

Register No.: **Name:**

Merge Sort

Write a Python program to sort a list of elements using the merge sort algorithm.

Program:

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = merge_sort(arr[:mid])
    right_half = merge_sort(arr[mid:])

    return merge(left_half, right_half)

def merge(left, right):
    sorted_arr = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            sorted_arr.append(left[i])
            i += 1
        else:
            sorted_arr.append(right[j])
            j += 1
    sorted_arr.extend(left[i:])
    sorted_arr.extend(right[j:])
    return sorted_arr
x = int(input("Enter the number of elements: "))
y = [int(i) for i in input("Enter the elements separated by spaces: ").split()]

sorted_list = merge_sort(y)
print("Sorted list:", sorted_list)
```

Input Format

The first line contains an integer,n , the size of the list a . The second line contains n, space-separated integers a[i].

Constraints

- 2<=n<=600
- 1<=a[i]<=$2x10^6$.

Output Format

You must print the following three lines of output:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

Sample Input 0

3

1 2 3

Sample Output 0

List is sorted in 0 swaps.

First Element: 1

Last Element: 3

For example:

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Ex. No. : 10.2 **Date:**

Register No.: **Name:**

Bubble Sort

Given an listof integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

For example, given a worst-case but small array to sort: a=[6,4,1]. It took 3 swaps to sort the array. Output would be

Array is sorted in 3 swaps.
First Element: 1
Last Element: 6

Program:

```
def bubble_sort(arr):
    n = len(arr)
    num_swaps = 0
    for i in range(n):
        for j in range(n - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                num_swaps += 1
    return num_swaps

n = int(input(""))
arr = list(map(int, input("").split()))

num_swaps = bubble_sort(arr)

print(f"List is sorted in {num_swaps} swaps.")
print(f"First Element: {arr[0]}")
print(f"Last Element: {arr[-1]}")
```

Input Format

The first line contains a single integer n , the length of A .
The second line contains n space-separated integers,A[i].

Output Format

Print peak numbers separated by space.

Sample Input

5

8 9 10 2 6

Sample Output

10 6

For example:

Input	Result
4 12 3 6 8	12 8

Ex. No. : 10.3 **Date:**

Register No.: **Name:**

Peak Element

Given an list, find peak element in it. A peak element is an element that is greater than its neighbors.

An element a[i] is a peak element if

A[i-1] <= A[i] >=a[i+1] for middle elements. [0<i<n-1]

A[i-1] <= A[i] for last element [i=n-1]

A[i]>=A[i+1] for first element [i=0]

Program:

```
n = int(input(""))
arr = list(map(int, input("").split()))

peaks = []

if n > 1 and arr[0] >= arr[1]:
    peaks.append(arr[0])

for i in range(1, n - 1):
    if arr[i - 1] <= arr[i] >= arr[i + 1]:
        peaks.append(arr[i])

if n > 1 and arr[-1] >= arr[-2]:
    peaks.append(arr[-1])

print(" ".join(map(str, peaks)))
```

For example:

Input	Result
1 2 3 5 8 6	False
3 5 9 45 42 42	True

Ex. No. : 10.4 **Date:**

Register No.: **Name:**

Binary Search

Write a Python program for binary search.

Program:

```
a=input()
b=[int(num) for num in a.split(",")]
c=int(input())
if c not in b:
    print("False")
else:
    print("True")
```

Input:

1 68 79 4 90 68 1 4 5

output:

1 2

4 2

5 1

68 2

79 1

90 1

For example:

Input	Result
4 3 5 3 4 5	3 2 4 2 5 2