

**Ex. No. : 9.2**

**Date:**

**Register No.: 230701353**

**Name: SURYA E**

---

### **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 i in range(n):
    data = input().split()
    key = data[0]
    values = list(map(int, data[1:]))
    test_dict[key] = values
sum_dict = {k: sum(v) for k, v in test_dict.items()}
sorted_sum_dict = dict(sorted(sum_dict.items(), key=lambda item: item[1]))
for key, value in sorted_sum_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 Johnny get maximum votes. Since John is alphabetically smaller, we print it. Use dictionary to solve the above problem

**Sample Input:**

```
10  
John  
John  
Johnny  
Jamie  
Jamie  
Johnny  
Jack  
Johnny  
Johnny  
Jackie
```

**Sample Output:**

Johnny

**For example:**

Input	Result
10 John John Johnny Jamie Jamie Johnny Jack Johnny Johnny Jackie	Johnny

**Ex. No. : 9.3**

**Date:**

**Register No.: 230701353**

**Name: SURYA E**

---

### **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:**

```
n = int(input())
counts = {}
for i in range(n):
    vote = input()
    counts[vote] = counts.get(vote, 0) + 1
max_votes = max(counts.values())
winners = [candidate for candidate, votes in counts.items() if votes == max_votes]
print(sorted(winners)[0])
```

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.: 230701353

Name: SURYA E

---

## **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 has 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:**

```
n = int(input())

student_data = {}

for i in range(n):

    data = input().split()

    name = data[0]

    test = int(data[1])

    assignment = int(data[2])

    lab = int(data[3])

    student_data[name] = (test, assignment, lab)

highest_average_students= []

highest_assignment_students = []

lowest_lab_students = []
```

```
lowest_avg_students = []

highest_avg = -1
highest_assignment = -1
lowest_lab = 101
lowest_avg = 101

for name, (test, assignment, lab) in student_data.items():

    avg_score = (test + assignment + lab) / 3

    if avg_score > highest_avg:

        highest_avg = avg_score

        highest_average_students = [name]

    elif avg_score == highest_avg:

        highest_average_students.append(name)

    if assignment > highest_assignment:

        highest_assignment = assignment

        highest_assignment_students = [name]

    elif assignment == highest_assignment:

        highest_assignment_students.append(name)
```

```
if lab < lowest_lab:

    lowest_lab = lab

    lowest_lab_students = [name]

elif lab == 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_average_students)))
print(" ".join(sorted(highest_assignment_students)))
print(" ".join(sorted(lowest_lab_students)))
print(" ".join(sorted(lowest_avg_students)))
```

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.: 230701353

Name: SURYA E

## **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 = {  
    '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 = input().upper()  
  
score = sum(points.get(letter, 0) for letter in word)  
  
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.: 230701353**

**Name: SURYA E**

### **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:
        # Finding the middle of the array
        mid = len(arr) // 2

        # Dividing the array elements into 2 halves
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
```

```
        arr[k] = right_half[j]
        j += 1
        k += 1

def main():
    n = int(input())
    elements = list(map(int, input().split()))

    merge_sort(elements)
    print(" ".join(map(str, elements)))

main()
```

### 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

- $2 \leq n \leq 600$
- $1 \leq a[i] \leq 2 \times 10^6$ .

### 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.: 230701353

Name: SURYA E

### **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 bubbleSort(arr):

    n = len(arr)

    num_swaps = 0

    for i in range (n):

        for j in range(0, n - i - 1):

            if arr[j] > arr[j + 1]:

                arr[j], arr[j + 1] = arr[j + 1], arr[j]

                num_swaps += 1

    print("List is sorted in", num_swaps, "swaps.")

    print("First Element:", arr[0])

    print("Last Element:", arr[-1])


def main():
```



```
n = int(input().strip())  
  
arr = list(map(int, input().strip().split()))  
  
bubbleSort(arr)  
  
main()
```

### 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.: 230701353**

**Name: SURYA E**

### **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

$A[i-1] \leq A[i] \geq A[i+1]$  for middle elements.  $[0 < i < n-1]$

$A[i-1] \leq A[i]$  for last element  $[i=n-1]$

$A[i] \geq A[i+1]$  for first element  $[i=0]$

#### **Program:**

```
def findPeakElements(arr):
    peaks = []
    n = len(arr)
    if 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 arr[n - 1] >= arr[n - 2]:
        peaks.append(arr[n - 1])
    return peaks

def main():
    n = int(input().strip())
    arr = list(map(int, input().strip().split()))
    peaks = findPeakElements(arr)
    print(" ".join(map(str, peaks)))
main()
```

**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.: 230701353**

**Name: SURYA E**

---

### **Binary Search**

Write a Python program for binary search.

**Program:**

```
def binary_search(arr, x):
    left = 0
    right = len(arr) - 1
    while left <= right:
        mid = left + (right - left) // 2
        if arr[mid] == x:
            return True
        elif arr[mid] < x:
            left = mid + 1
        else:
            right = mid - 1
    return False

def main():
    arr = list(map(int, input().strip().split(',')))
    x = int(input().strip())
    result = binary_search(sorted(arr), x)
    print(result)
main()
```



**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

**Ex. No. : 10.5**

**Date:**

**Register No.: 230701353**

**Name: SURYA E**

---

### **Frequency of Elements**

To find the frequency of numbers in a list and display in sorted order.

**Constraints:**

$1 \leq n$ ,  $\text{arr}[i] \leq 100$

**Program:**

```
input_numbers = input().strip().split()
numbers = [int(x) for x in input_numbers]
frequency = {}
for number in numbers:
    if number in frequency:
        frequency[number] += 1
    else:
        frequency[number] = 1
sorted_numbers = sorted(frequency.keys())
for number in sorted_numbers:
    print(number, frequency[number])
```