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.

```
n = int(input())
array = input().split()
for i in range(n):
    array[i] = int(array[i])
for i in range(n):
    swapped = False
    for j in range(0, n - i - 1):
        if array[j] > array[j + 1]:
            array[j], array[j + 1] = array[j + 1], array[j]
            swapped = True
    if not swapped:
        break

for i in range(n):
    print(array[i], end=' ')
print()
```

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
```

```
def bubble_sort(arr):

    n = len(arr)

    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]

                swaps += 1


    return swaps
```

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

# Sort and count swaps
num_swaps = bubble_sort(arr)

# Output
print("List is sorted in", num_swaps, "swaps.")
print("First Element:", arr[0])
print("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]

```
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.

```
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.: **Name:**

Frequency of Elements

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

Constraints:

1<=n, arr[i]<=100

```
num=input()
num=num.split()
numbers=[]
for i in num:
    numbers.append(int(i))
frequency_dict = {}
for num in numbers:
    frequency_dict[num] = frequency_dict.get(num, 0) + 1
sorteds = {k: v for k, v in sorted(frequency_dict.items())}
for num, freq in sorteds.items():
    print(num,freq)
```