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.

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]
    left_sorted = merge_sort(left_half)
    right_sorted = merge_sort(right_half)
    return merge(left_sorted, right_sorted)
def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    while i < len(left):
        result.append(left[i])
        i += 1
    while j < len(right):
        result.append(right[j])
        j += 1
    return result
n = int(input())
arr = list(map(int, input().split()))
sorted_arr = merge_sort(arr)
print(" ".join(map(str, sorted_arr)))
```

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 bubbleSort(arr):

    n = len(arr)

    numSwaps = 0



    for i in range(n):

        swapped = False



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

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

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

                numSwaps += 1

                swapped = True
        if not swapped:
            break

    print("List is sorted in", numSwaps, "swaps.")
    print("First Element:", arr[0])
    print("Last Element:", arr[-1])

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

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]

```
def find_peak_element(nums):
    if not nums:
        return None  # If the list is empty, return None
    if len(nums) == 1:
        return nums[0]  # If the list has only one element, that element is the peak

    n = len(nums)

    for i in range(n):
        if (i == 0 and nums[i] >= nums[i + 1])
            return nums[i]

    return None
```

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.

```
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

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

```
def frequencySorted(arr):
    freq_dict = {}
    for num in arr:
        if num in freq_dict:
            freq_dict[num] += 1
        else:
            freq_dict[num] = 1
    sorted_freq = sorted(freq_dict.items())
    for key, value in sorted_freq:
        print(key, value)
arr = list(map(int, input().strip().split()))
frequencySorted(arr)
```