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.: 230701104 **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:
        # Find the middle of the array
        mid = len(arr) // 2
        L = arr[:mid]
        R = arr[mid:]
        merge_sort(L)
        merge_sort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1
if __name__ == "__main__":
    n = int(input())
    arr = list(map(int, input().split()))
    merge_sort(arr)
    print(*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
```

```
a=int(input())
count=0
b=[int(x) for x in input().split()]
for j in range(a):
    for i in range(a-j-1):
        if(b[i]>b[i+1]):
            count+=1
            b[i],b[i+1]=b[i+1],b[i]
print("List is sorted in",count,"swaps.")
print("First Element:",b[0])
print("Last Element:",b[-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())
A = list(map(int, input().split()))
peaks = []
if n > 0 and (n == 1 or A[0] >= A[1]):
    peaks.append(A[0])
for i in range(1, n-1):
    if A[i-1] <= A[i] >= A[i+1]:
        peaks.append(A[i])
if n > 1 and A[n-1] >= A[n-2]:
    peaks.append(A[n-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.

```
a = input()
b = [int(i) for i in a.split(',')]
b.sort()

m = int(input())
first = 0
last = len(b) - 1
flag = 0

while first <= last:
    mid = (first + last) // 2
    if b[mid] == m:
        flag = 1
        break
    elif b[mid] < m:
        first = mid + 1
    else:
        last = mid - 1

if flag:
    print("True")
else:
    print("False")
```

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

```
a=[int(x) for x in input().split()]
a.sort()
b={}
for i in a:
    if i not in b:
        b[i]=1
    else:
        b[i]+=1
for i in b:
    print(i,b[i])
```