10 - Searching & Sorting

Ex. No. : 10.1 **Date: 01.06.24**

Register No.: 231901020 **Name: KAVIYA.V**

Merge Sort

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

CODING:

```
def ms(arr):
    if len(arr)>1:
        mid=len(arr)//2
        l=arr[:mid]
        r=arr[mid:]
        ms(l)
        ms(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(j<len(r)):
        arr[k]=r[j]
        j+=1
        k+=1
    while(i<len(l)):
        arr[k]=l[i]
        i+=1
        k+=1
n=int(input())
arr=list(map(int,input().split()))
ms(arr)
print(*arr)
```

	Input	Expected	Got	
✔	5 6 5 4 3 8	3 4 5 6 8	3 4 5 6 8	✔
✔	9 14 46 43 27 57 41 45 21 70	14 21 27 41 43 45 46 57 70	14 21 27 41 43 45 46 57 70	✔
✔	4 86 43 23 49	23 43 49 86	23 43 49 86	✔

For example:

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

Ex. No. : 10.2 **Date: 01.06.24**

Register No.: 231901020 **Name: KAVIYA.V**

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

CODING:

```
n = int(input())
a = [int(x) for x in input().split()]
num_swaps = 0
for i in range(n):
    for j in range(0, n-i-1):
        if a[j] > a[j+1]:
            a[j], a[j+1] = a[j+1], a[j]
            num_swaps += 1
print(f'List is sorted in {num_swaps} swaps.")
print(f'First Element: {a[0]}")
print(f'Last Element: {a[-1]}")
```

	Input	Expected	Got	
✔	6 3 4 8 7 1 2	1 2 3 4 7 8	1 2 3 4 7 8	✔
✔	6 9 18 1 3 4 6	1 3 4 6 9 18	1 3 4 6 9 18	✔
✔	5 4 5 2 3 1	1 2 3 4 5	1 2 3 4 5	✔

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. : 10.3 **Date: 01.06.24**

Register No.: 231901020 **Name: KAVIYA.V**

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]

CODING:

```
n = int(input())
A = [int(x) for x in input().split()]
peak_elements = []
if n > 1 and A[0] >= A[1]:
    peak_elements.append(A[0])
elif n == 1:
    peak_elements.append(A[0])
for i in range(1, n-1):
    if A[i] >= A[i-1] and A[i] >= A[i+1]:
        peak_elements.append(A[i])
if n > 1 and A[n-1] >= A[n-2]:
    peak_elements.append(A[n-1])
print(" ".join(map(str, peak_elements)))
```

	Input	Expected	Got	
✔	7 15 7 10 8 9 4 6	15 10 9 6	15 10 9 6	✔
✔	4 12 3 6 8	12 8	12 8	✔

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. : 10.4 **Date: 01.06.24**

Register No.: 231901020 **Name: KAVIYA.V**

Binary Search

Write a Python program for binary search.

CODING:

```
def bs(arr,x):
    arr.sort()
    l,r=0,len(arr)-1
    while l<=r:
        m=(l+r)//2
        if arr[m]==x:
            return True
        elif arr[m]<x:
            l=m+1
        else:
            r=m-1
    return False
n=list(map(int,input().split(',')))
target=int(input())
result=bs(n,target)
```

```
print(result)
```

For example:

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

Ex. No. : 10.5 **Date:**

Register No.: 231901020 **Name: KAVIYA.V**

Frequency of Elements

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

Constraints:

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

CODING:

```
n=list(map(int,input().split()))
f={}
for i in n:
    f[i]=f.get(i,0)+1
sf=sorted(f.items())
for i,f in sf:
    print(i,f)
```

	Input	Expected	Got	
✔	4 3 5 3 4 5	3 2 4 2 5 2	3 2 4 2 5 2	✔
✔	12 4 4 4 2 3 5	2 1 3 1 4 3 5 1 12 1	2 1 3 1 4 3 5 1 12 1	✔
✔	5 4 5 4 6 5 7 3	3 1 4 2 5 3 6 1 7 1	3 1 4 2 5 3 6 1 7 1	✔

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