**10 - Searching & Sorting**

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

**Register No.: 231801024 Name: Boopesh.R**

**Merge Sort**

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

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  6 5 4 3 8 | 3 4 5 6 8 |

**PROGRAM:**

def bubble\_sort(arr):

n = len(arr)

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\_elements = int(input())

array = list(map(int, input().split()))

bubble\_sort(array)

for element in array:

print(element, end=" ")

**OUTPUT:**

****

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

**Register No.: 231801024 Name: Boopesh.R**

**Bubble Sort**

Given anlistof integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1.      [List](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "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](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "List).

3.      Last Element: lastElement, the *last* element in the sorted [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "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

**Input Format**

The first line contains an integer,n , the size of the [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "List) a .  
The second line contains  n,  space-separated integers a[i].

**Constraints**

·         2<=n<=600

·         1<=a[i]<=2x106.

**Output Format**

You must print the following three lines of output:

1.      [List](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "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](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "List).

3.      Last Element: lastElement, the *last* element in the sorted [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "List).

**Sample Input 0**

3

1 2 3

**Sample Output 0**

[List](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "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 |

**PROGRAM:**

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

n = int(input())

arr = list(map(int, input().split()))

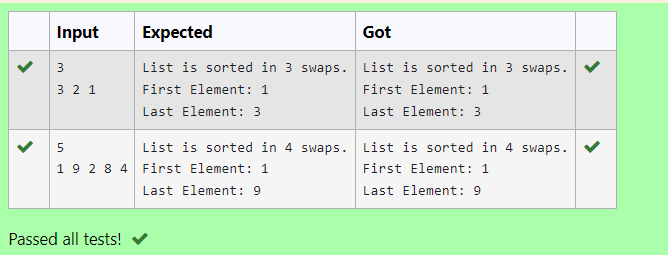
num\_swaps = bubble\_sort(arr)

print("List is sorted in", num\_swaps, "swaps.")

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

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

**OUTPUT:**



**Ex. No. : 10.3 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**Peak Element**

Given an [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068" \o "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]

**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 |

**PROGRAM:**

def find\_peaks(nums):

peaks = []

for i in range(len(nums)):

if i == 0:

if nums[i] >= nums[i+1]:

peaks.append(nums[i])

elif i == len(nums) - 1:

if nums[i] >= nums[i-1]:

peaks.append(nums[i])

else:

if nums[i] >= nums[i-1] and nums[i] >= nums[i+1]:

peaks.append(nums[i])

return peaks

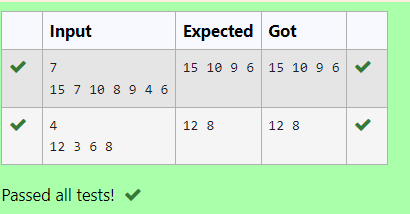
n = int(input())

nums = list(map(int, input().split()))

peaks = find\_peaks(nums)

print(' '.join(map(str, peaks)))

**OUTPUT:**

**Ex. No. : 10.4 Date:**

**Register No.: 231801024 Name: Boopesh.R**

**Binary Search**

Write a Python program for binary search.

**For example:**

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

**PROGRAM:**

def binary\_search(arr, x):

arr.sort()

left, right = 0,len(arr) - 1

while left <= right:

mid=(left + right) // 2

if arr[mid]== x:

return True

elif arr[mid] < x:

left=mid + 1

else:

right=mid-1

return False

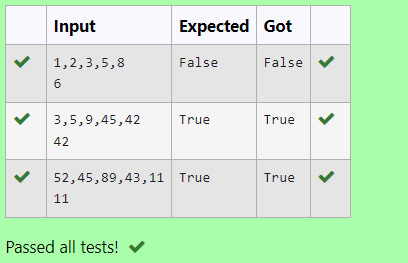
numbers = list(map(int, input().split(',')))

target=int(input())

result=binary\_search(numbers, target)

print(result)

**OUTPUT:**



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

**Register No.: 231801024 Name: Boopesh.R**

**Frequency of Elements**

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

**Constraints:**

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

**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 |

**PROGRAM:**

arr = list(map(int, input().split()))

frequency\_dict = {}

for num in arr:

if num in frequency\_dict:

frequency\_dict[num] += 1

else:

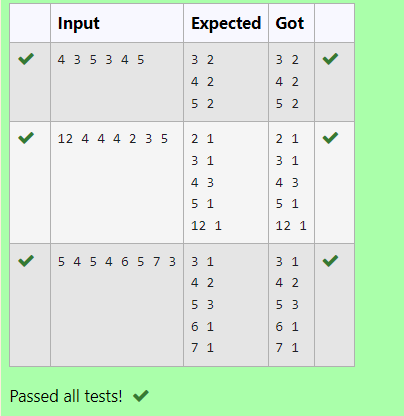
frequency\_dict[num] = 1

sorted\_frequency = sorted(frequency\_dict.items())

for key, value in sorted\_frequency:

print(key, value)

**OUTPUT:**

****