10 - Searching & Sorting

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.

For example:

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

```
Def merge_sort(arr):
  If len(arr) <= 1:
    Return arr
  Mid = len(arr) // 2
  Left_half = arr[:mid]
  Right_half = arr[mid:]
  Left_half = merge_sort(left_half)
  Right_half = merge_sort(right_half)
  Return merge(left_half, right_half)


Def merge(left, right):
  Merged = []
  Left_index = right_index = 0


  While left_index < len(left) and right_index < len(right):
    If left[left_index] < right[right_index]:
      Merged.append(left[left_index])
      Left_index += 1
    Else:
      Merged.append(right[right_index])
      Right_index += 1
  Merged.extend(left[left_index:])
  Merged.extend(right[right_index:])


  Return merged


N = int(input())
Arr = list(map(int, input().split()))
Sorted_arr = merge_sort(arr)
Print(*sorted_arr)
```

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
```

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]<=2x10⁶.

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

```
#BUBBLE SORT
n=int(input())
s=input()
l=s.split()
l=[int(l[i]) for i in range(0,len(l))]
c=0
for i in range(0,n):
  for j in range(0,n-i-1):
    if l[j]>l[j+1]:
      l[j],l[j+1]=l[j+1],l[j]
      c=c+1
print("List is sorted in",c,"swaps.")
print("First Element:",l[0])
print("Last Element:",l[-1])
```

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]

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

```
#peak element
n=int(input())
s=input()
z=s.split()
l=[]

for i in range(0,n):
   if i==0:
      if int(z[i])>int(z[i+1]):
         l.append(int(z[i]))
      else:
         l.append(int(z[i+1]))
   elif i==n-1:
      if int(z[i])>int(z[i-1]) and int(z[i]) not in l:
         l.append(int(z[i]))
      elif int(z[i])<int(z[i-1]) and int(z[i-1]) not in l:
         l.append(int(z[i-1]))
   else:
      m=int(z[i-1])
      for j in range(i-1,i+2):
         if m<int(z[j]):
            m=int(z[j])
      if m not in l:
         l.append(m)

for i in l:
   print(i,end=' ')
```

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

Register No.: **Name:**

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

```
#binary search
s = input().split(',')
s = [int(i) for i in s]
n = int(input())
f = 0
mid = s[len(s)-1]
low = s[0]
high = s[len(s)-1]
if(mid==n):
   f=1
if(f==0):
   while(low!=mid and high!=mid):
      if(mid<n):
         low = s[mid+1]
         mid = [(low+high)//2]
      elif(mid>n):
         high = s[mid-1]
         mid = [(low+high)//2]
      else:
         f = 1
         break
if(f==1):
   print(True)
else:
   print(False)
```

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

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

```
S = input()
Z = s.split()
Z = [int(z[i]) for I in range(len(z))]
z.sort()
l = list()
for I in range(0,len(z)):
   c=1
   for j in range(i+1,len(z)):
      if z[i]==z[j]:
         c=c+1
   if z[i] not in l:
      print(z[i],c,end=' ')
      l.append(z[i])
      print()
```