10 - Searching & Sorting

Ex. No. : 10.1 **Date: 1/06/2024**

Register No.: 231401007 **Name: Ananya Sriram**

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:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]
        merge_sort(left_half)
        merge_sort(right_half)
        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
```

```
            j += 1
        k += 1
    while i < len(left_half):
        arr[k] = left_half[i]
        i += 1
        k += 1
    while j < len(right_half):
        arr[k] = right_half[j]
        j += 1
        k += 1
n = int(input())
arr = list(map(int, input().split()))
merge_sort(arr)
for num in arr:
    print(num, end=" ")
```

	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	✔

Passed all tests! ✔

Ex. No. : 10.2 **Date: 1/06/2024**

Register No.: 231401007 **Name: Ananya Sriram**

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

- $2<=n<=600$
- $1<=a[i]<=2x10^6$.

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

```
def bubble_sort(arr):
    n = len(arr)
    num_swaps = 0
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                # Swap the elements
                arr[j], arr[j+1] = arr[j+1], arr[j]
                num_swaps += 1
                swapped = True
        if not swapped:
            break
    return num_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])
```

	Input	Expected	Got	
✔	3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3	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	List is sorted in 4 swaps. First Element: 1 Last Element: 9	✔

Passed all tests! ✔

```
def find_peak_elements(arr):
    n = len(arr)
    peak_elements = []
    if n == 1:
        return arr
    if arr[0] >= arr[1]:
        peak_elements.append(arr[0])
    for i in range(1, n - 1):
        if arr[i] >= arr[i - 1] and arr[i] >= arr[i + 1]:
            peak_elements.append(arr[i])
    if arr[n - 1] >= arr[n - 2]:
        peak_elements.append(arr[n - 1])
    return peak_elements
n = int(input())
arr = list(map(int, input().split()))
peak_elements = find_peak_elements(arr)
print(*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	✔

Passed all tests! ✔

Ex. No. : 10.4 **Date: 1/06/2024**

Register No.: 231401007 **Name: Ananya Sriram**

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, target): left, right = 0, len(arr) - 1 while left <= right:

mid = (left + right) // 2 if arr[mid] == target:

return True

elif arr[mid] < target: left = mid + 1

else:

right = mid - 1 return False

arr_input = input() target_input = input()

arr = list(map(int, arr_input.split(','))) target = int(target_input)

	Input	Expected	Got	
	1,2,3,5,86	False	False	
	3,5,9,45,4242	True	True	
	52,45,89,43,1111	True	True	

Ex. No. : 10.5 **Date: 1/06/2024**

Register No.: 231401007 **Name: Ananya Sriram**

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

```
def frequency_count(arr):
    frequency_dict = {}
    for num in arr:
        if num in frequency_dict:
            frequency_dict[num] += 1
        else:
```

```
            frequency_dict[num] = 1
    return frequency_dict
arr = list(map(int, input().split()))
freq_dict = frequency_count(arr)
sorted_freq = sorted(freq_dict.items())
for key, value in sorted_freq:
    print(key, value)
```

	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	✔

Passed all tests! ✔