Started on	Friday, 7 June 2024, 10:04 PM
State	Finished
Completed on	Friday, 7 June 2024, 10:21 PM
Time taken	16 mins 51 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. You read a list of numbers. You need to arrange the elements in ascending order and print the result. The sorting should be done using bubble sort.

Input Format: The first line reads the number of elements in the array. The second line reads the array elements one by one.

Output Format: The output should be a sorted list.

For example:

Input	Result
6 3 4 8 7 1 2	1 2 3 4 7 8
5 4 5 2 3 1	1 2 3 4 5

Answer: (penalty regime: 0 %)

```python
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]

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

print(*arr)
```

	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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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

Answer: (penalty regime: 0 %)

```
def find_peak_elements(arr):
    n = len(arr)
    peaks = []

    if n > 1 and arr[0] >= arr[1]:
        peaks.append(arr[0])

    for i in range(1, n - 1):
        if arr[i - 1] <= arr[i] >= arr[i + 1]:
            peaks.append(arr[i])

    if n > 1 and arr[-1] >= arr[-2]:
        peaks.append(arr[-1])

    return peaks

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	✓

Question **3**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```python
def binary_search(arr,x):
    arr.sort()
    left,right=0,len(arr)-1
    while left<=right:
        m=(left+right)//2
        if arr[m]==x:
            return True
        elif arr[m]<x:
            left=m+1
        else:
            right=m-1
    return False
n=list(map(int,input().split(',')))
t=int(input())
r=binary_search(n,t)
print(r)
```

	Input	**Expected**	**Got**	
✓	1,2,3,5,8 6	False	False	✓
✓	3,5,9,45,42 42	True	True	✓
✓	52,45,89,43,11 11	True	True	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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

Answer: (penalty regime: 0 %)

```
def bubble_sort(arr):
    n = len(arr)
    num_swaps = 0
    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_swaps += 1
    return num_swaps, arr[0], arr[-1]
if __name__=="__main__":
    n= int(input().strip())
```

	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! ✓

Correct

Marks for this submission: 1.00/1.00.

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`

Answer: (penalty regime: 0 %)

```
arr = list(map(int,input().split()))
def count_frequency(arr):
    freq_dict = {}
    for num in arr:
        freq_dict[num] = freq_dict.get(num, 0) + 1
    return freq_dict
freq_dict= count_frequency(arr)
sorted_freq = sorted(freq_dict.items())
for num,freq in sorted_freq:
    print(num,freq)
```

	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`
				✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week10_MCQ

Jump to...

Sorting ►