

<b>Started on</b>	Saturday, 1 June 2024, 5:25 PM
<b>State</b>	Finished
<b>Completed on</b>	Saturday, 1 June 2024, 5:28 PM
<b>Time taken</b>	3 mins 18 secs
<b>Marks</b>	5.00/5.00
<b>Grade</b>	<b>100.00</b> out of 100.00

# Question 1

Correct

Mark 1.00 out of 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

- $2 \leq n \leq 600$
- $1 \leq a[i] \leq 2 \times 10^6$ .

## 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 %)

```
1 n = int(input())
2 a = list(map(int, input().split()))
3
4 swaps = 0
```

```

4 swaps = 0
5 for i in range(n):
6     for j in range(n-1):
7         if a[j] > a[j+1]:
8             a[j], a[j+1] = a[j+1], a[j]
9             swaps += 1
10
11 print(f"List is sorted in {swaps} swaps.")
12 print(f"First Element: {a[0]}")
13 print(f>Last Element: {a[-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! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

An [list](#) contains N numbers and you want to determine whether two of the numbers sum to a given number K. For example, if the input is 8, 4, 1, 6 and K is 10, the answer is yes (4 and 6). A number may be used twice.

**Input Format**

The first line contains a single integer n , the length of [list](#)

The second line contains n space-separated integers, [list\[i\]](#).

The third line contains integer k.

**Output Format**

Print Yes or No.

**Sample Input**

```
7
0 1 2 4 6 5 3
1
```

**Sample Output**

```
Yes
```

**For example:**

Input	Result
5 8 9 12 15 3 11	Yes
6 2 9 21 32 43 43 1 4	No

**Answer:** (penalty regime: 0 %)

```
1 n = int(input())
2 arr = list(map(int, input().split()))
3 k = int(input())
4 s = set()
5
6 for num in arr:
7
8     if k - num in s:
9         print("Yes")
10        break
11    s.add(num)
12 else:
13
14     print("No")
15
```

	Input	Expected	Got	
✓	5 8 9 12 15 3 11	Yes	Yes	✓
✓	6 2 9 21 32 43 43 1 4	No	No	✓
✓	6 13 42 31 4 8 9 17	Yes	Yes	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



## Question 3

Correct

Mark 1.00 out of 1.00

Bubble Sort is the simplest [sorting](#) algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. You read an [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 %)

```

1 a = int(input())
2 b = input().split()
3 c=[]
4 d=[]
5 for i in b:
6     if int(i) < 10:
7         c.append(i)
8     else:
9         d.append(i)
10
11 c.sort()
12 d.sort()
13 e =c+d
14 print(*e)
15

```

	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.

## Question 4

Correct

Mark 1.00 out of 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

$A[i-1] \leq A[i] \geq a[i+1]$  for middle elements.  $[0 < i < n-1]$

$A[i-1] \leq A[i]$  for last element  $[i=n-1]$

$A[i] \geq A[i+1]$  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 %)

```
1 n = int(input().strip())
2 arr = list(map(int, input().strip().split()))
3
4 peaks = []
5
6 if n > 0 and (n == 1 or arr[0] >= arr[1]):
7     peaks.append(arr[0])
8
9 for i in range(1, n - 1):
10     if arr[i] >= arr[i - 1] and arr[i] >= arr[i + 1]:
11         peaks.append(arr[i])
12
13 if n > 1 and arr[n - 1] >= arr[n - 2]:
14     peaks.append(arr[n - 1])
15
16 print(" ".join(map(str, peaks)))
17
```



	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! ✓

Correct

Marks for this submission: 1.00/1.00.

## Question 5

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 %)

```

1 arr = list(map(int, input().split(',')))
2 x = int(input())
3 arr.sort()
4 low = 0
5 high = len(arr) - 1
6 found = False
7
8 while low <= high:
9     mid = (low + high) // 2
10    if arr[mid] < x:
11        low = mid + 1
12    elif arr[mid] > x:
13        high = mid - 1
14    else:
15        found = True
16        break
17
18 print(found)
19

```

	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.

◀ Week10\_MCQ

Jump to...

Sorting ▶

