State	Finished
Completed on	Wednesday, 19 June 2024, 9:24 PM
Time taken	33 mins 4 secs
Marks	3.00/5.00
Grade	**60.00** out of 100.00

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.

Input	Result
1,2,3,5,8 6	False
3,5,9,45,42 42	True

Answer: (penalty regime: 0 %)

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 %)

```python
def find_peak_elements(arr):
    n = len(arr)

    # Edge case: if array has less than 2 elements, return empty list
    if n <= 1:
        return []

    peaks = []

    # Check for peak elements in the middle of the array
    for i in range(1, n - 1):
        if arr[i - 1] <= arr[i] >= arr[i + 1]:
            peaks.append(arr[i])

    # Check the first element
    if arr[0] >= arr[1]:
        peaks.append(arr[0])

    # Check the last element
    if arr[n - 1] >= arr[n - 2]:
        peaks.append(arr[n - 1])

    return peaks

# Input reading and processing
n = int(input().strip())
arr = list(map(int, input().strip().split()))

# Finding the peak elements
peak_elements = find_peak_elements(arr)

# Output the peak elements separated by space
print(" ".join(map(str, peak_elements)))

```

	12 3 6 8			

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

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

Answer: (penalty regime: 0 %)

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Divide the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Recursively sort each half
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    # Merge the sorted halves
    sorted_arr = merge(left_half, right_half)

    return sorted_arr

def merge(left, right):
    sorted_arr = []
    left_idx, right_idx = 0, 0

    # Merge left and right sublists into sorted_arr
    while left_idx < len(left) and right_idx < len(right):
        if left[left_idx] <= right[right_idx]:
            sorted_arr.append(left[left_idx])
            left_idx += 1
        else:
            sorted_arr.append(right[right_idx])
            right_idx += 1

    # Append remaining elements
    sorted_arr.extend(left[left_idx:])
    sorted_arr.extend(right[right_idx:])

    return sorted_arr

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

sorted_arr = merge_sort(arr)
print(*sorted_arr)

```

	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	✓

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 %)

```python
def find_sum_pair(nums, K):
    # Set to store seen numbers
    seen = set()

    # Traverse through the list
    for num in nums:
        # Calculate the complement
        complement = K - num

        # Check if complement exists in the set
        if complement in seen:
            return "Yes"

        # Add current number to set
        seen.add(num)

    # If no pair found
    return "No"

# Input reading and processing
n = int(input().strip())
nums = list(map(int, input().strip().split()))
K = int(input().strip())

# Check for sum pair
result = find_sum_pair(nums, K)

# Output result
print(result)

```

	11			
✓	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.

◄ Week10_MCQ

Jump to...

Sorting ►