WEEK 10

1. 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

**Program:**

n =int(input())

numbers = list(map(int, input().split()))

K =int(input())

for i in range(n):

for j in range (i + 1, n) :

if numbers[i] + numbers[j] == K:

print("Yes")

exit()

print("No")

**Output:**

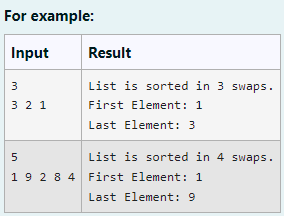
****

2. 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.



**Program:**

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 arr, num\_swaps

n = int(input().strip())

a = list(map(int, input().strip().split()))

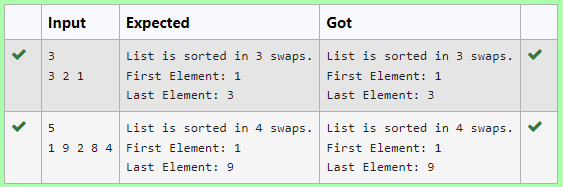
sorted\_array, num\_swaps = bubble\_sort(a)

print(f"List is sorted in {num\_swaps} swaps.")

print(f"First Element: {sorted\_array[0]}")

print(f"Last Element: {sorted\_array[-1]}")

**Output:**

****

3. 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

**Program:**

def find\_peak\_elements(arr):

n = len(arr)

peaks = []

if n > 0 and (n == 1 or 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[n-1] >= arr[n-2]:

peaks.append(arr[n-1])

return peaks

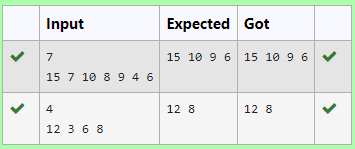
n = int(input().strip())

a = list(map(int, input().strip().split()))

peak\_elements = find\_peak\_elements(a)

print(" ".join(map(str, peak\_elements)))

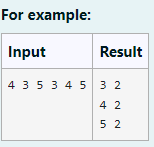
**Output:**

****

4. To find the frequency of numbers in a list and display in sorted order.

**Constraints:**

1<=n, arr[i]<=100



**Program:**

def find\_frequencies(arr):

frequency = {}

for num in arr:

if num in frequency:

frequency[num] += 1

else:

frequency[num] = 1

sorted\_frequency = dict(sorted(frequency.items()))

return sorted\_frequency

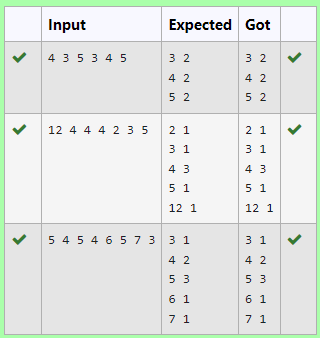
arr = list(map(int, input().strip().split()))

sorted\_frequency = find\_frequencies(arr)

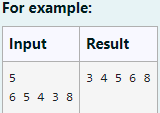
for num, count in sorted\_frequency.items():

print(f"{num} {count}")

**output:**

****

5. Write a Python program to sort a list of elements using the merge sort algorithm.



**Program:**

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr)

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

return arr

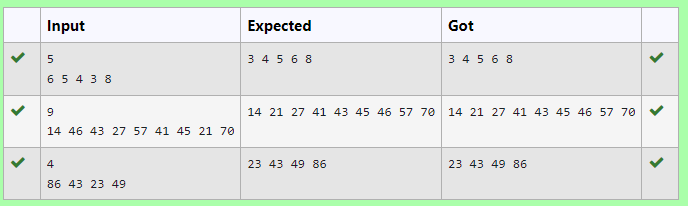
n = int(input().strip())

a = list(map(int, input().strip().split()))

sorted\_array = merge\_sort(a)

print(" ".join(map(str, sorted\_array)))

**Output:**

****