# WEEK-10

1.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 |

**PROGRAM:**

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

frequency = {}

for num in nums:

frequency[num] = frequency.get(num, 0) + 1

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

for num, freq in sorted\_frequency:

print(num, freq)

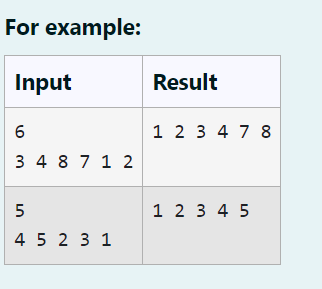
**OUTPUT:**

****

2.) 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.



**Program:**

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

swapped = True

if not swapped:

break

return arr

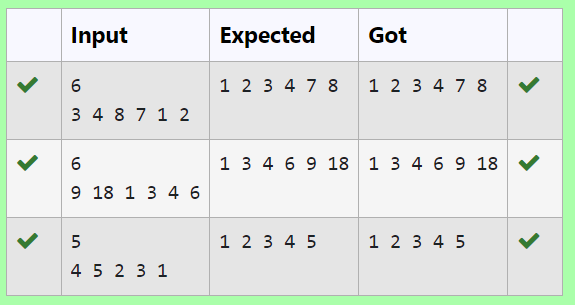
n = int(input())

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

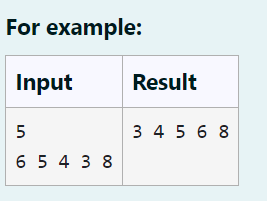
sorted\_arr = bubble\_sort(arr)

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

**OUTPUT**



3. 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)//2

l=arr[:mid]

r=arr[mid:]

merge\_sort(l)

merge\_sort(r)

i=j=k=0

while i<len(l) and j<len(r):

if l[i]<r[j]:

arr[k]=l[i]

i+=1

else:

arr[k]=r[j]

j+=1

k+=1

while i<len(l):

arr[k]=l[i]

i+=1

k+=1

while j<len(r):

arr[k]=r[j]

j+=1

k+=1

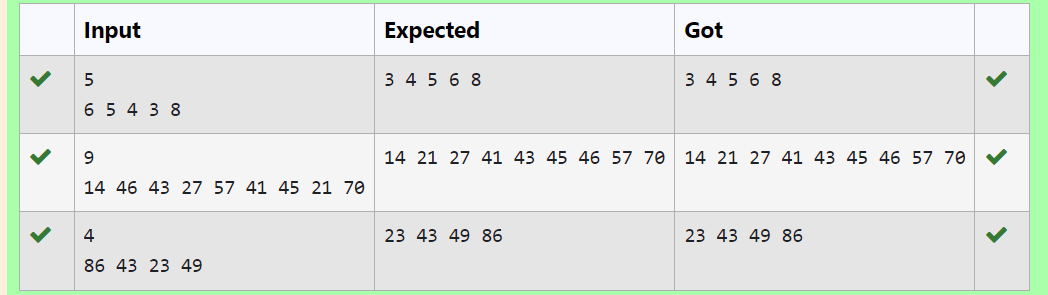
n=int(input())

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

merge\_sort(arr)

print(\*arr)

**OUTPUT:**



4. 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):

arr.sort()

l, r = 0, len(arr) - 1

while l <= r:

mid = (l + r) // 2

if arr[mid] == target:

return True

elif arr[mid] < target:

l = mid + 1

else:

r = mid - 1

return False

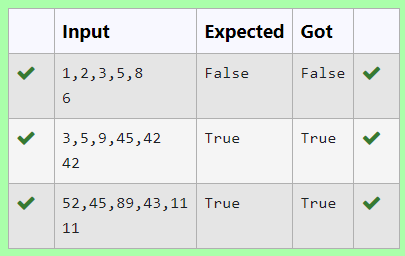
num = list(map(int, input().split(',')))

x = int(input())

result = binary\_search(num, x)

print(result)

**OUTPUT:**



5**.**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

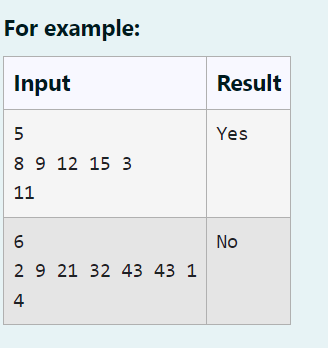
7

0 1 2 4 6 5 3

1

Sample OUTPUT

Yes

****

**PROGRAM:**

def check\_sum\_exists(arr, k):

seen = set()

for num in arr:

if k - num in seen:

return "Yes"

seen.add(num)

return "No"

# Input

n = int(input())

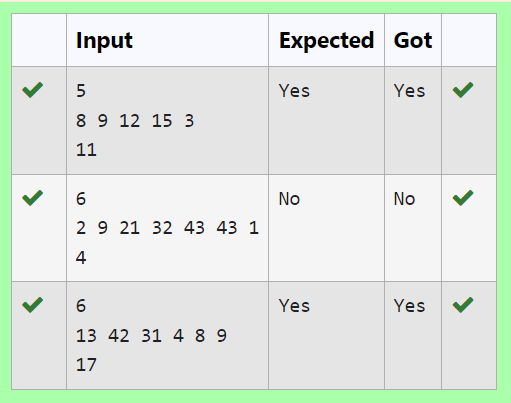
arr = list(map(int, input().split()))

k = int(input())

# OUTPUT

print(check\_sum\_exists(arr, k))

**OUTPUT:**

****