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()))

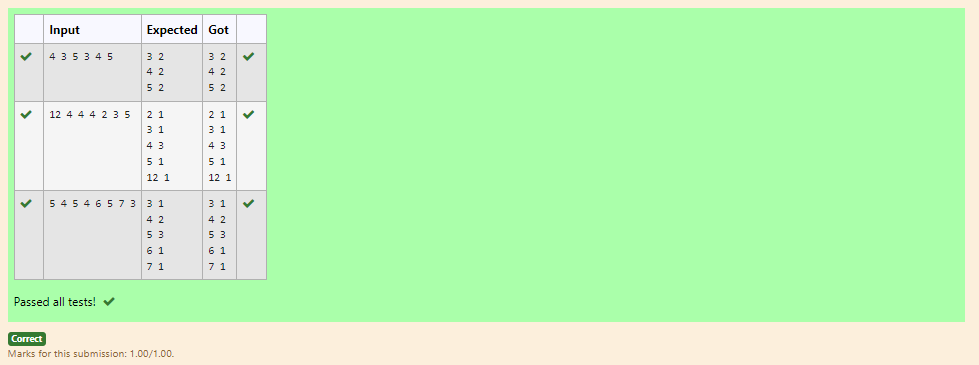
frequency={}

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)

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

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  6 5 4 3 8 | 3 4 5 6 8 |

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]

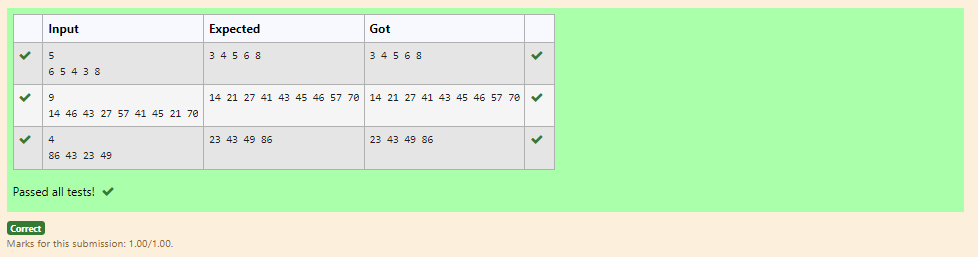
j+=1

k+=1

n=int(input())

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

merge\_sort(arr)

print(\*arr)

3 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 |

Program:

n=int(input())

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

k=int(input())

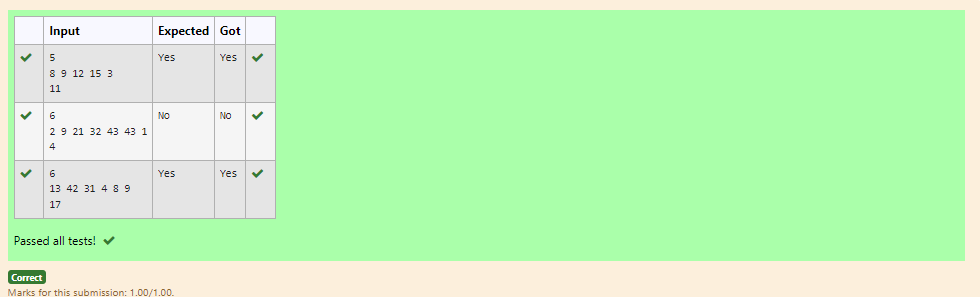
for i in range(n):

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

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

print("Yes")

exit()

print("No")

4 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]<=2x106.

**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 |

Program:

def bubble\_sort(arr):

n = len(arr)

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]

swaps += 1

return swaps

n = int(input())

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

num\_swaps = bubble\_sort(arr)

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

print("First Element:", arr[0])

print("Last Element:", arr[-1])

5.  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

# Check if target is present at mid

if arr[mid] == target:

return True

# If target is greater, ignore left half

elif arr[mid] < target:

l = mid + 1

# If target is smaller, ignore right half

else:

r = mid - 1

# If the target is not present in the array

return False

# Get input from the user

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

x = int(input())

# Perform binary search

result = binary\_search(num,x)

# Print True or False based on search result

print(result) 