**Week10: LINEAR AND BINARY**

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.

**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):

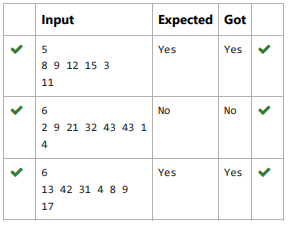
if numbers[i]+numbers[j]==K:

print("Yes")

exit()

print("No")

**OUTPUT:**

****

2. 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.

**PROGRAM:**

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()))

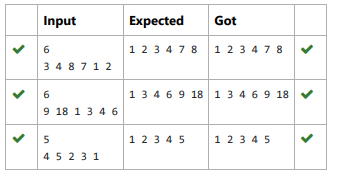
# Perform bubble sort

bubble\_sort(arr)

# Output

print(" ".join(map(str,arr)))

**OUTPUT:**

****

3. 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

**PROGRAM:**

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

frequency={}

for num in numbers:

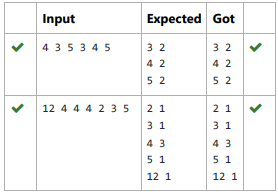
frequency[num]=frequency.get(num,0)+1

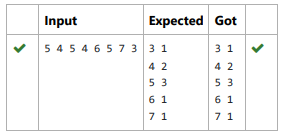
sorted\_frequency=sorted(frequency.items())

for num,freq in sorted\_frequency:

print(num,freq)

**OUTPUT:**

****

****

4. 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

**Sample Input:**

5

8 9 10 2 6

**Sample Output**

10 6

**PROGRAM:**

# Function to find peak elements

def find\_peak(arr):

n = len(arr)

peaks = []

# Check for the first element

if n > 1 and arr[0] >= arr[1]:

peaks.append(arr[0])

# Check for the middle elements

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

if arr[i - 1] <= arr[i] >= arr[i + 1]:

peaks.append(arr[i])

# Check for the last element

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

peaks.append(arr[n - 1])

return peaks

# Input

n = int(input())

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

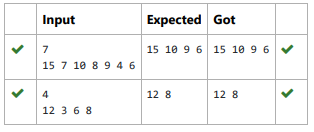
# Find peaks

peak\_values = find\_peak(arr)

# Output

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

**OUTPUT:**

****

5. 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

**Sample Input 0**

3

1 2 3

**Sample Output 0**

List is sorted in 0 swaps.

First Element: 1

Last Element: 3

**PROGRAM:**

def bubble\_sort(arr):

num\_swaps = 0

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]

num\_swaps+=1

swapped=True

if not swapped:f not swapped:

break

return num\_swaps

n=int(input())

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

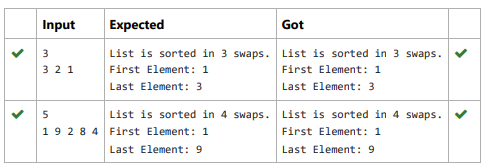
num\_swaps=bubble\_sort(arr)

print("List is sorted in", num\_swaps,"swaps.")

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

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

**OUTPUT:**

****