1. 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

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| 4 12 3 6 8 | 12 8 |

def find\_peak\_elements(arr):

n = len(arr)

peak\_elements = []

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

peak\_elements.append(arr[0])

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

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

peak\_elements.append(arr[i])

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

peak\_elements.append(arr[-1])

return peak\_elements

n = int(input())

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

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

print(\*peak\_elements)

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.

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 |

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 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])

3. 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.

**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 |

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

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

def print\_list(arr):

for element in arr:

print(element, end=" ")

print()

x=int(input())

y=[int(i) for i in input().split()]

merge\_sort(y)

for i in y:

print(i,end=" ")

4. 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 |

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

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

def print\_list(arr):

for element in arr:

print(element, end=" ")

print()

x=int(input())

y=[int(i) for i in input().split()]

merge\_sort(y)

for i in y:

print(i,end=" ")

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 |

array=input()

l=list(map(int,array.split(",")))

num=int(input())

low=0

high=len(l)-1

f=1

for i in range(len(l)):

swapped=False

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

if(l[j]>l[j+1]):

l[j],l[j+1]=l[j+1],l[j]

swapped=True

if(swapped==False):

break

while(low<=high):

mid=(low+high)//2

if(num==l[mid]):

f=0

break

elif(num<l[mid]):

high=mid-1

else:

low=mid+1

if(f):

print("False")

else:

print("True")