10 - Searching & Sorting

**Ex. No. : 10.1 Date: 25/5/2024**

**Register No: 231501030 Name: Benin.A.F**

# Merge Sort

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:**

a=int(input()) l=[]

l.extend(input().split()) for i in range(a-1):

for j in range(a-1): if(int(l[j])>int(l[j+1])):

t=int(l[j]) l[j]=int(l[j+1]) l[j+1]=t

for i in range(a): print(int(l[i]),end=" ")



**Ex. No. : 10.2 Date: 25/5/2024**

**Register No: 231501030 Name: Benin.A.F**

# Bubble Sort

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 ex[am](http://118.185.187.137/moodle/mod/resource/view.php?id=1068)ple, 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]: # Swap elements

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

return swaps

# Input the size of the list n = int(input())

# Input the list of integers

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

# Perform bubble sort and count the number of swaps num\_swaps = bubble\_sort(arr)

# Print the number of swaps

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

# Print the first element print("First Element:", arr[0])

# Print the last element print("Last Element:", arr[-1])



**Ex. No. : 10.3 Date: 25/5/2024**

**Register No: 231501030 Name: Benin.A.F**

# Peak Element

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 |

**PROGRAM:**

def find\_peak(arr): peak\_elements = []

# Check for the first element if arr[0] >= arr[1]:

peak\_elements.append(arr[0])

# Check for middle elements for i in range(1, len(arr) - 1):

if arr[i - 1] <= arr[i] >= arr[i + 1]: peak\_elements.append(arr[i])

# Check for the last element if arr[-1] >= arr[-2]:

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

# Input the length of the list n = int(input())

# Input the list of integers

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

# Find peak elements and print the result peak\_elements = find\_peak(arr) print(\*peak\_elements)



**Ex. No. : 10.4 Date: 25/5/2024**

**Register No: 231501030 Name: Benin.A.F**

# Binary Search

Write a Python program for binary search.

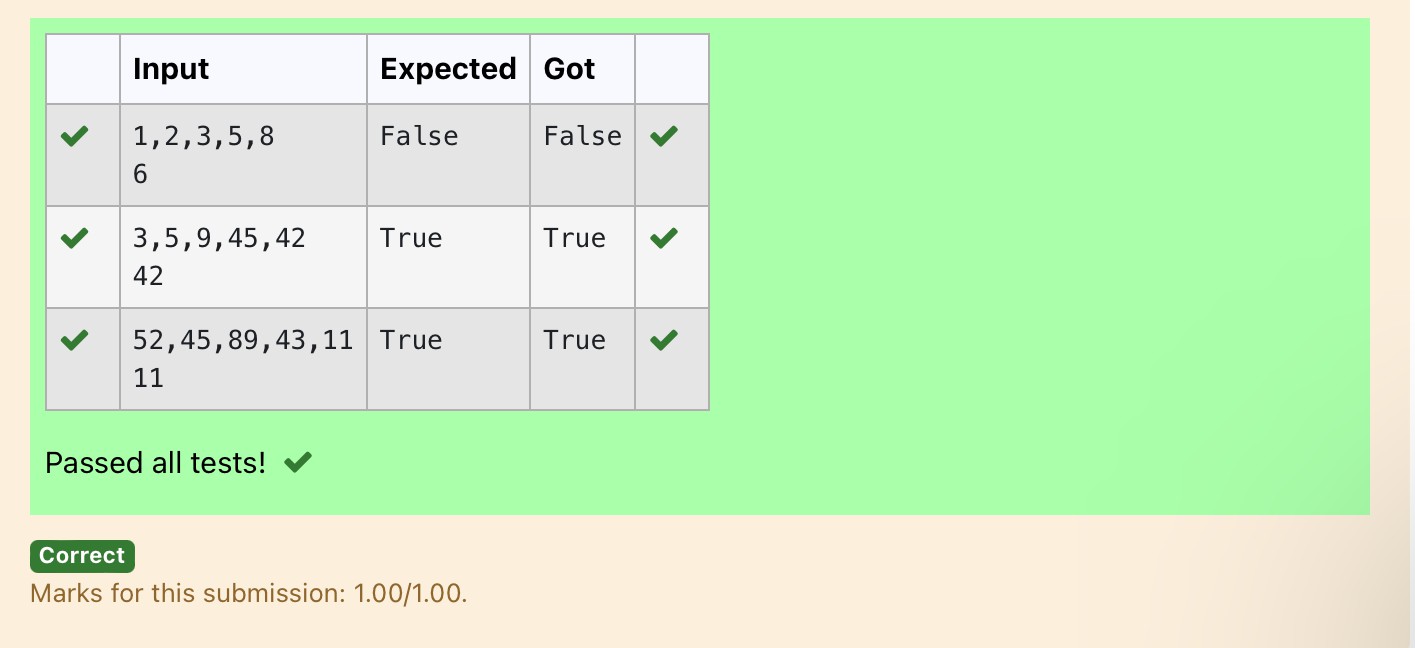
**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| 1 2 3 5 8  6 | False |
| 3 5 9 45 42  42 | True |

**PROGRAM:**

a = input().split(",")

b = input() print(b in a)



**Ex. No. : 10.5 Date: 25/5/2024**

**Register No: 231501030 Name: Benin.A.F**

# Frequency of Elements

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:**

def count\_frequency(arr): frequency = {}

# Count the frequency of each number in the list for num in arr:

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

# Sort the dictionary based on keys sorted\_frequency = sorted(frequency.items())

# Print the frequency of each number for num, freq in sorted\_frequency:

print(num, freq)

# Input the list of numbers

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

# Count the frequency and print the result count\_frequency(arr)

