

Input	Result
5 6 5 4 3 8	3 4 5 6 8

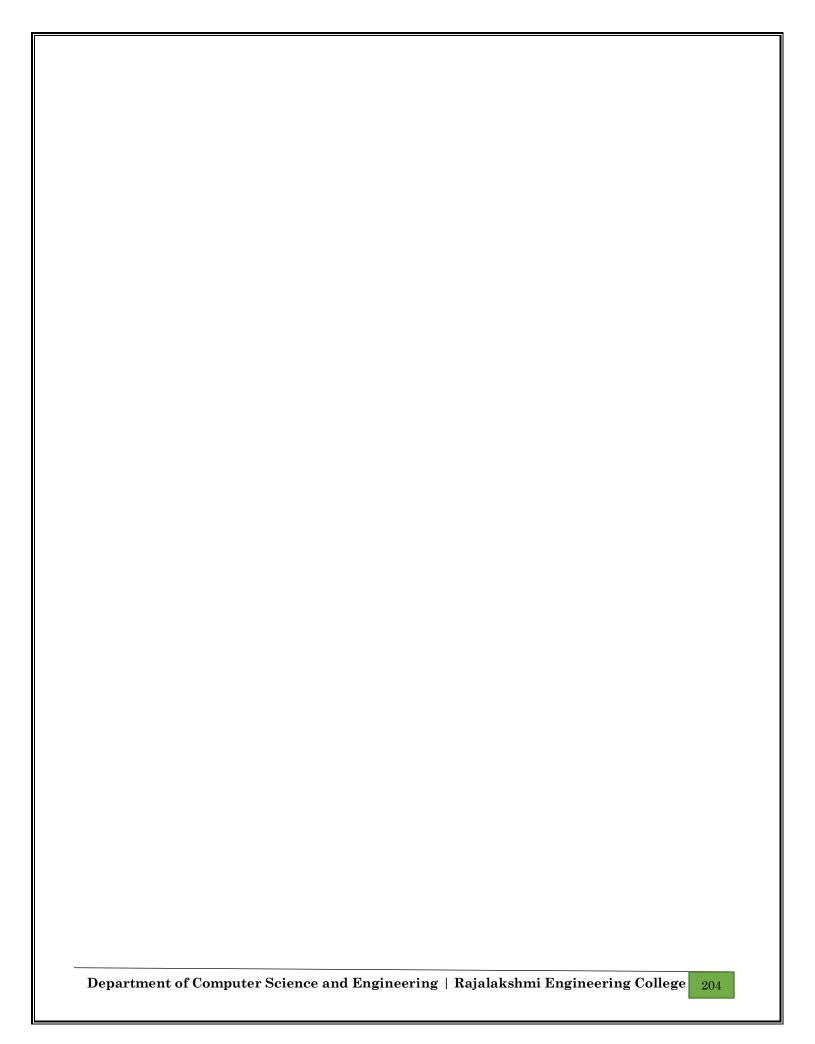
Ex. No.	:	10.1	Date:
Register No	.:		Name:

# Merge Sort

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

### PROGRAM:

```
aa=int(input())
bb=input()
cc=list(map(int,bb.split()))
ccc=sorted(cc)
for j in range(0,len(ccc)):
    print(ccc[j],end=" ")
```



#### **Input Format**

The first line contains an integer, n, the size of the <u>list</u> a. The second line contains n, space-separated integers a[i].

#### **Constraints**

- · 2<=n<=600
- $1 \le a[i] \le 2x10^6$ .

#### **Output Format**

You must print the following three lines of output:

- 1. <u>List</u> 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 <u>list</u>.
- 3. Last Element: lastElement, the *last* element in the sorted list.

#### Sample Input 0

3

123

#### Sample Output 0

<u>List</u> is sorted in 0 swaps.

First Element: 1

Last Element: 3

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 19284	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Ex. No. : 10.2 Date:

Register No.: Name:

## **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. <u>List</u> 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 <u>list</u>.
- 3. Last Element: lastElement, the *last* element in the sorted <u>list</u>.

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

#### PROGRAM:

```
n=int(input())
num=input()
num=num.split()
arr=[] count=0
for i in num:
    arr.append(int(i))
for i in range(n-1):
    for j in range(0, n-i-1):
        if arr[j] > arr[j + 1]:
            arr[j], arr[j + 1] = arr[j + 1], arr[j]
            count+=1
print("List issorted in",count,"swaps.")
print("First Element:",arr[0])
print("Last Element:",arr[n-1])
```

### **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

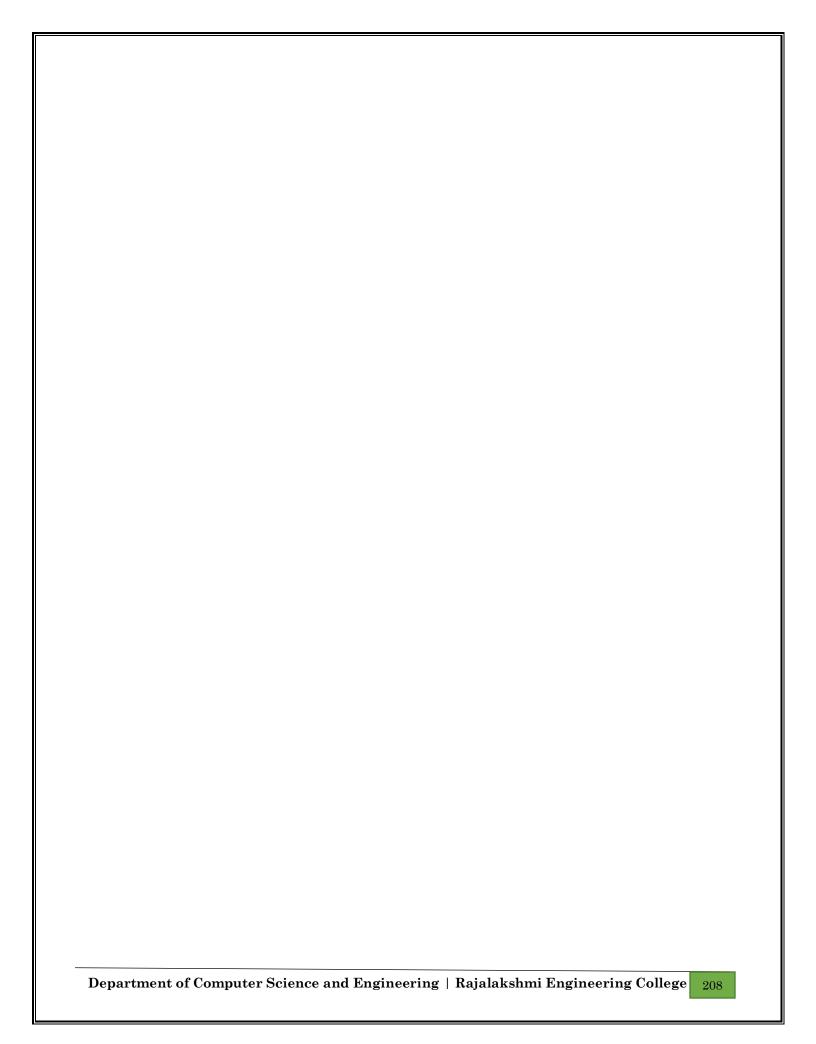
5

891026

### Sample Output

10 6

Input	Result
4 12 3 6 8	12 8



Ex. No. : 10.3 Date:

Register No.: Name:

## **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] \le A[i] \ge a[i+1] for middle elements. [0 \le i \le n-1]
A[i-1] \le A[i] for last element [i=n-1]
A[i] > = A[i+1] for first element [i=0]
           PROGRAM:
                a=int(input())
                aa=input()
                b=list(map(int,aa.split()))
                for i in range(0,len(b)):
                   if(i==0):
                     if((b[i])>(b[i+1])):
                        print(b[i],end=" ")
                   elif(i==(len(b)-1)):
                     if((b[i])>(b[i-1])):
                        print(b[i],end=" ")
                   elif(i>0 \text{ and } i<len(b)-1):
                     if((b[i]>b[i+1]) \text{ and } ((b[i])>(b[i-1]))):
                        print(b[i],end=" ")
```

Input	Result
12358	False
3 5 9 45 42 42	True

Ex. No. : 10.4 Date:

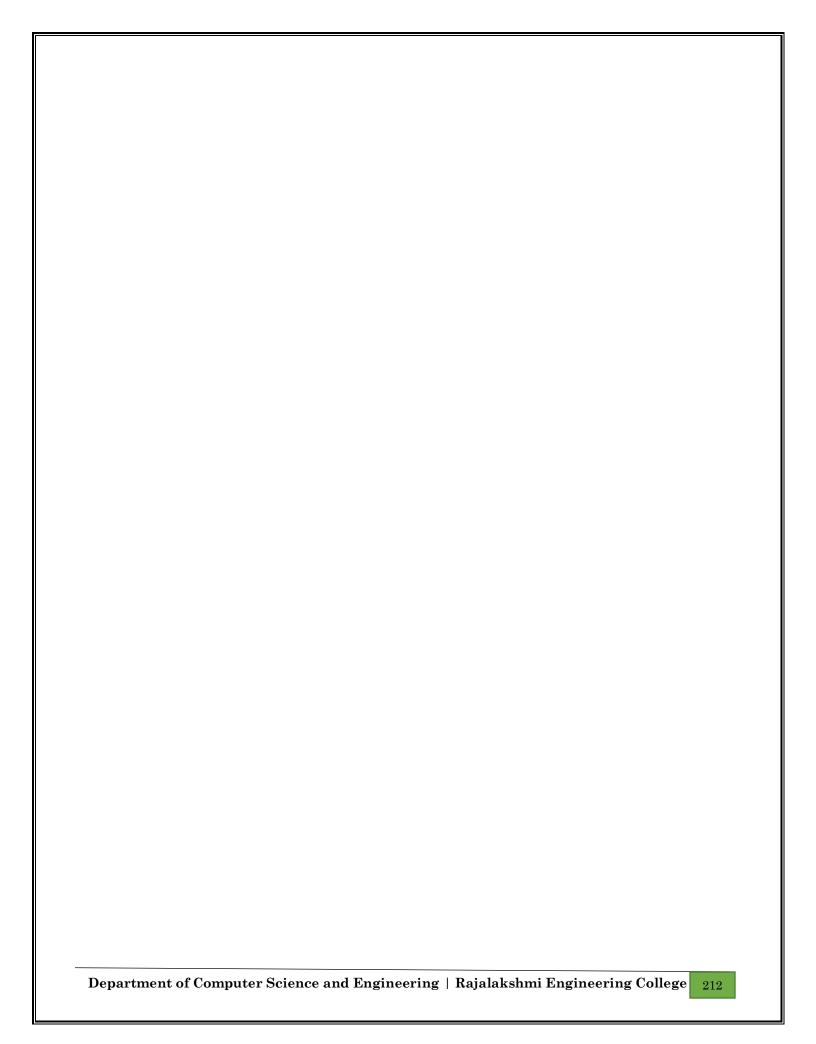
Register No.: Name:

# **Binary Search**

Write a Python program for binary search.

### PROGRAM:

```
bb=input()
a=list(map(int,bb.split(",")))
bbb=int(input())
if(bbb in a):
   print("True")
else:
   print("False")
```



# Input:

1 68 79 4 90 68 1 4 5

## output:

12

4 2

5 1

68 2

79 1

90 1

Input	Result
4 3 5 3 4 5	3 2 4 2 5 2

Ex. No. : 10.5 Date:

Register No.: Name:

# **Frequency of Elements**

To find the frequency of numbers in a list and display in sorted order.

#### **Constraints:**