

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

Ex. No. : 10.1 Date: 01.06.24

Register No.: 230701334 Name: SREYA G

Merge Sort

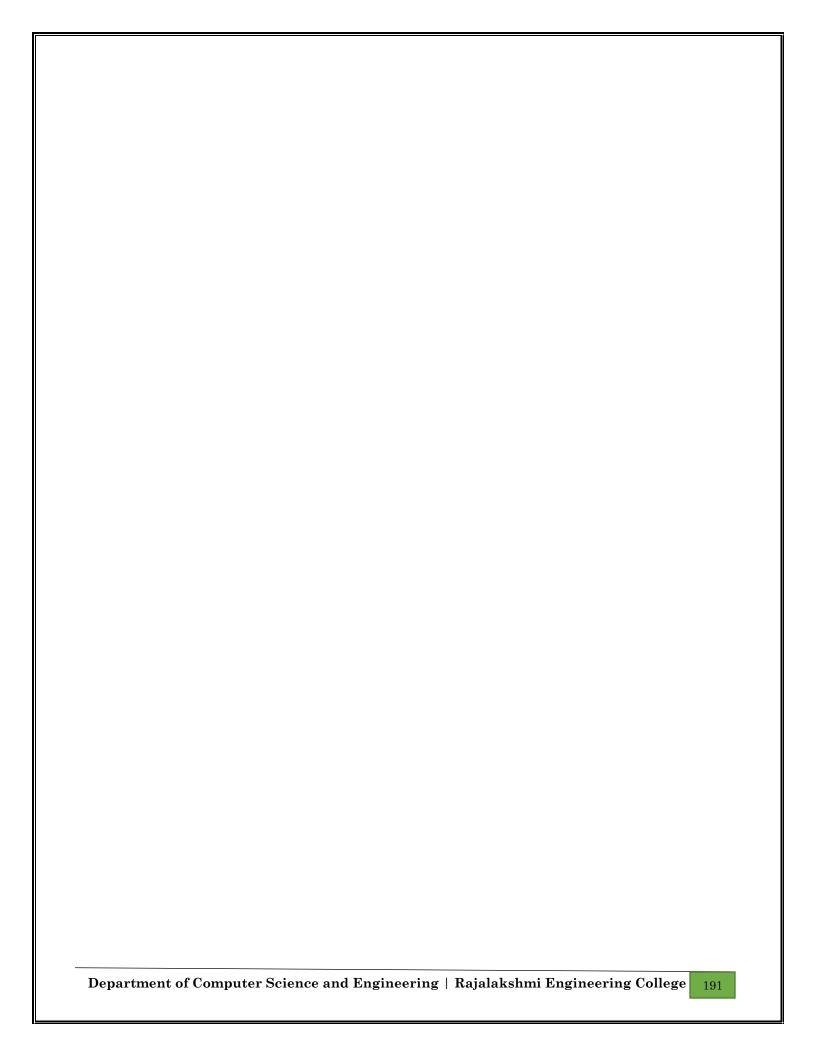
Write a Python program to sort a list of elements using the merge sort algorithm.

```
def ms(arr):
  if len(arr) > 1:
     mid = len(arr)//2
     left_half = arr[:mid]
     right_half = arr[mid:]
     ms(left_half)
     ms(right_half)
     i = j = k = 0
     while i< len(left_half) and j< len(right_half):
       if left_half[i] < right_half[j]:</pre>
          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

n=int(input())
arr = []
arr= [int(e) for e in input().split()]
ms(arr)

for i in range(len(arr)):
    print(arr[i],end=" ")</pre>
```



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 <u>list</u>.

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: 01.06.24

Register No.: 230701334 Name: SREYA G

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:

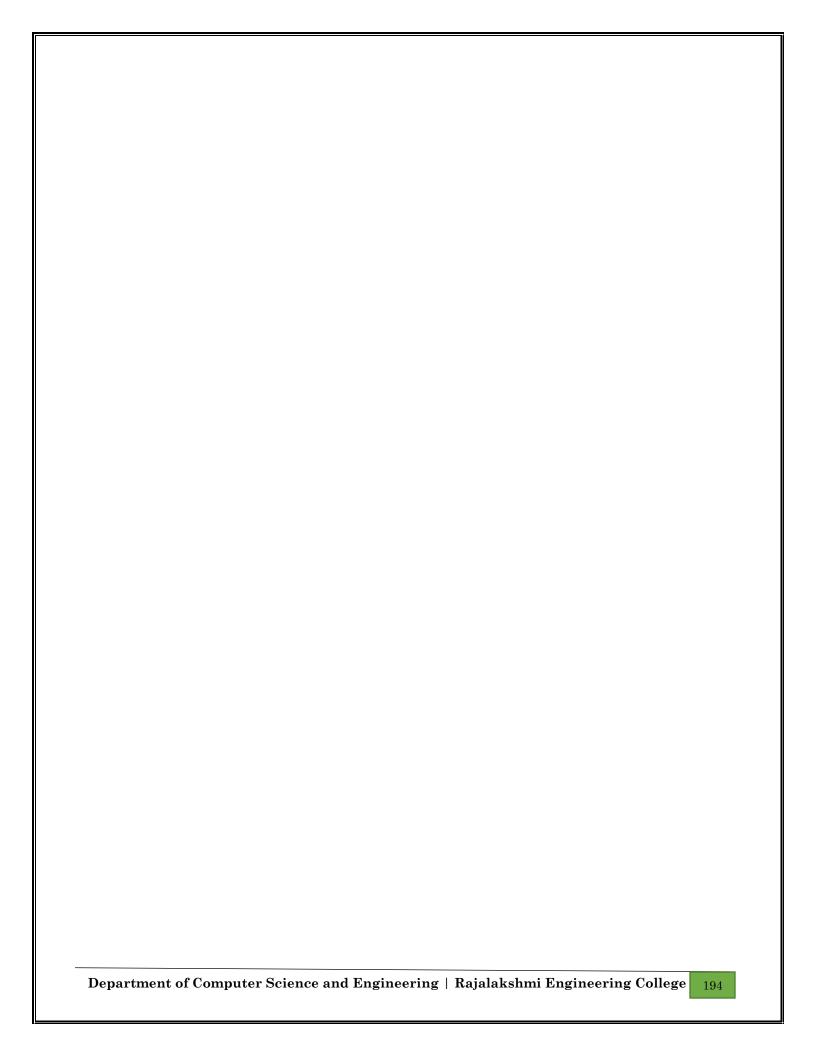
- 1. <u>List</u> is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
- 2. First Element: firstElement, the *first* element in the sorted list.
- 3. Last Element: lastElement, the *last* element in the sorted list.

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

```
def bubbleSort(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
  print("List is sorted in", num_swaps, "swaps.")
  print("First Element:", arr[0])
  print("Last Element:", arr[-1])
def main():
  n = int(input().strip())
  arr = list(map(int, input().strip().split()))
  bubbleSort(arr)
main()
```



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

106

or example:		
Input	Result	
4 12 3 6 8	12 8	

Ex. No. : 10.3 Date: 01.06.24

Register No.: 230701334 Name: SREYA G

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] >= a[i+1] for middle elements. [0<i<n-1] A[i-1] \le A[i] for last element [i=n-1] A[i] >= A[i+1] for first element [i=0]
```

```
n=int(input())
lst=input().split()
lst=[int(e) for e in lst]
if lst[0]>lst[1]:
    print(lst[0],end=" ")
for i in range(1,n-2):
    if lst[i]>lst[i-1] and lst[i]>lst[i+1]:
        print(lst[i],end=" ")
if lst[-1]>lst[-2]:
    print(lst[-1])
```

Input	Result
12358	False
3 5 9 45 42 42	True

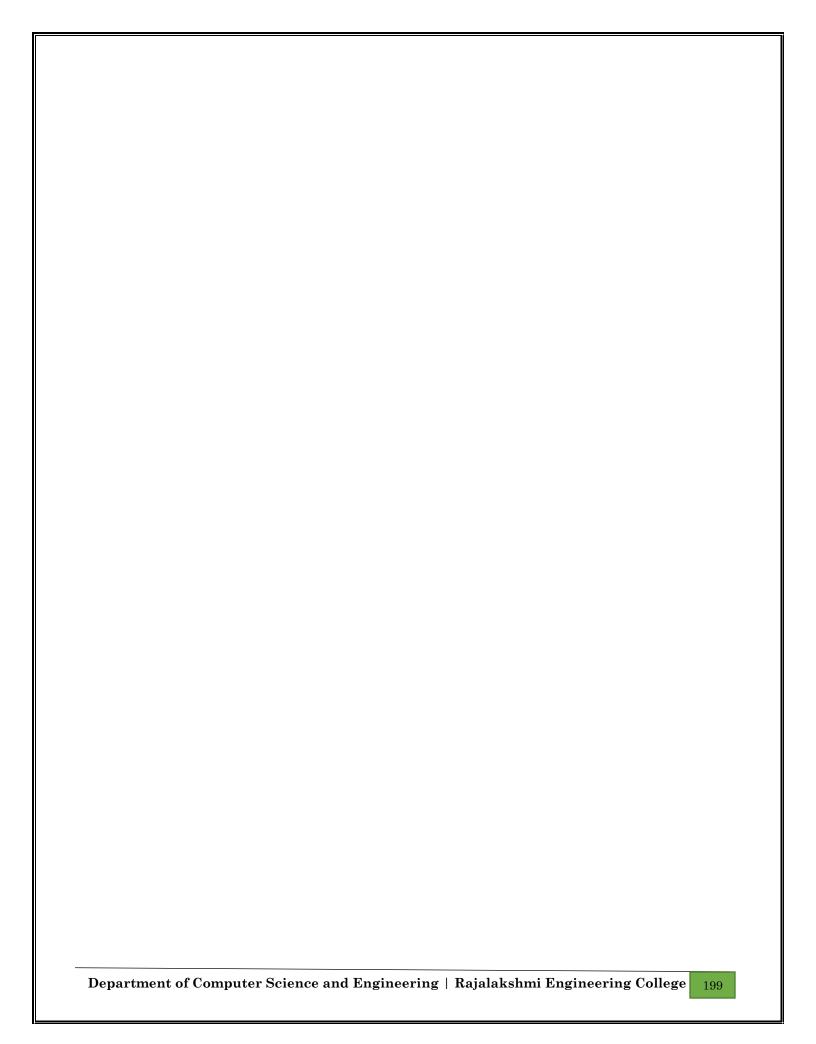
Ex. No. : 10.4 Date: 01.06.24

Register No.: 230701334 Name: SREYA G

Binary Search

Write a Python program for binary search.

```
lst=input().split(',')
for i in range(len(lst)):
  lst[i] = int(lst[i])
search = int(input())
def BinarySearch(l,k):
  flag = 0
  low = 0
  high = len(lst)
  while low<=high:
     mid = low + (high-low)//2
    if k==l[mid]:
       return "True"
     elif k>l[mid]:
       low = mid+1
     else:
       high = mid-1
  return "False"
print(BinarySearch(sorted(lst),search))
```



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: 01.06.24

Register No.: 230701334 Name: SREYA G

Frequency of Elements

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

Constraints:

```
1<=n, arr[i]<=100
```

```
lst=input().split()
lst=[int(e) for e in lst]
lst.sort()
dup=[]
for i in lst:
    count=0
    for y in lst:
        if i==y:
            count+=1
    if i not in dup:
        print(i,count)
        dup.append(i)
```