

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

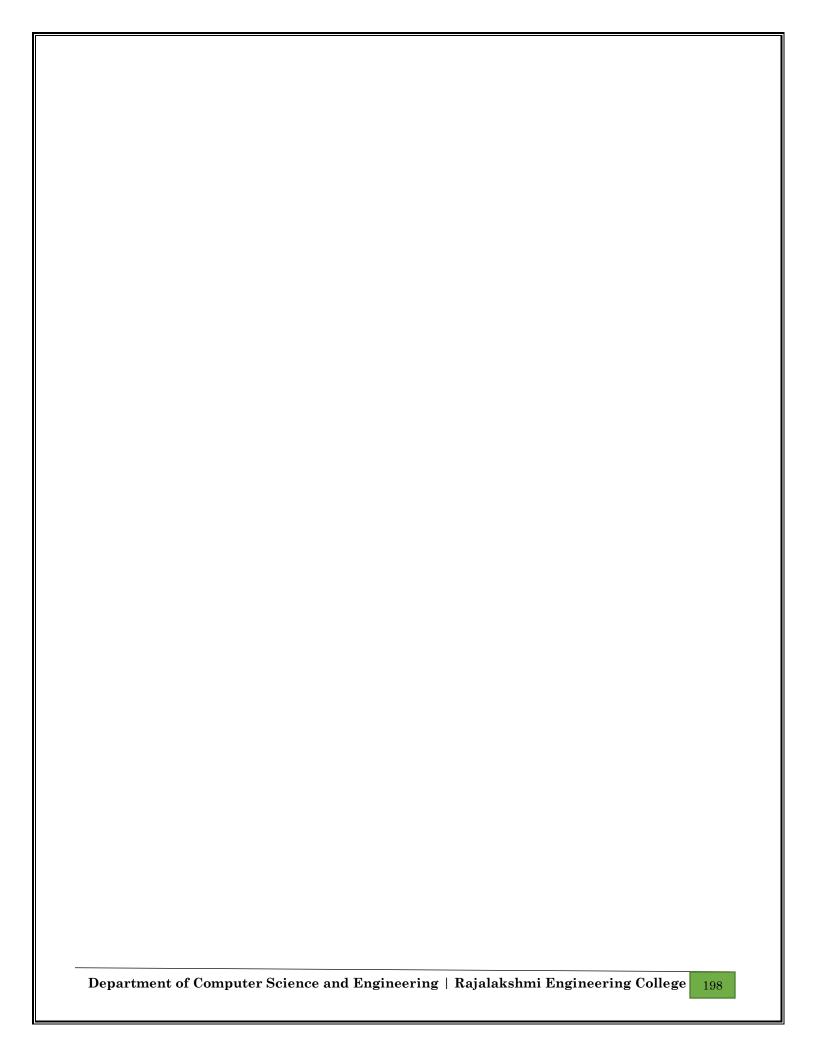
Ex. No. : 10.1 Date:

Register No.: Name:

Merge Sort

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

```
n = int(input())
array = input().split()
for i in range(n):
  array[i] = int(array[i])
for i in range(n):
  swapped = False
  for j in range(0, n - i - 1):
     if array[j] > array[j + 1]:
       array[j], array[j + 1] = array[j + 1], array[j]
       swapped = True
  if not swapped:
     break
for i in range(n):
  print(array[i], end=' ')
print()
```



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
- $1 <= a[i] <= 2x10^6$.

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:

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

a=int(input())
count=0
b=[int(x) for x in input().split()]
for j in range(a):
 for i in range(a-j-1):
 if(b[i]>b[i+1]):

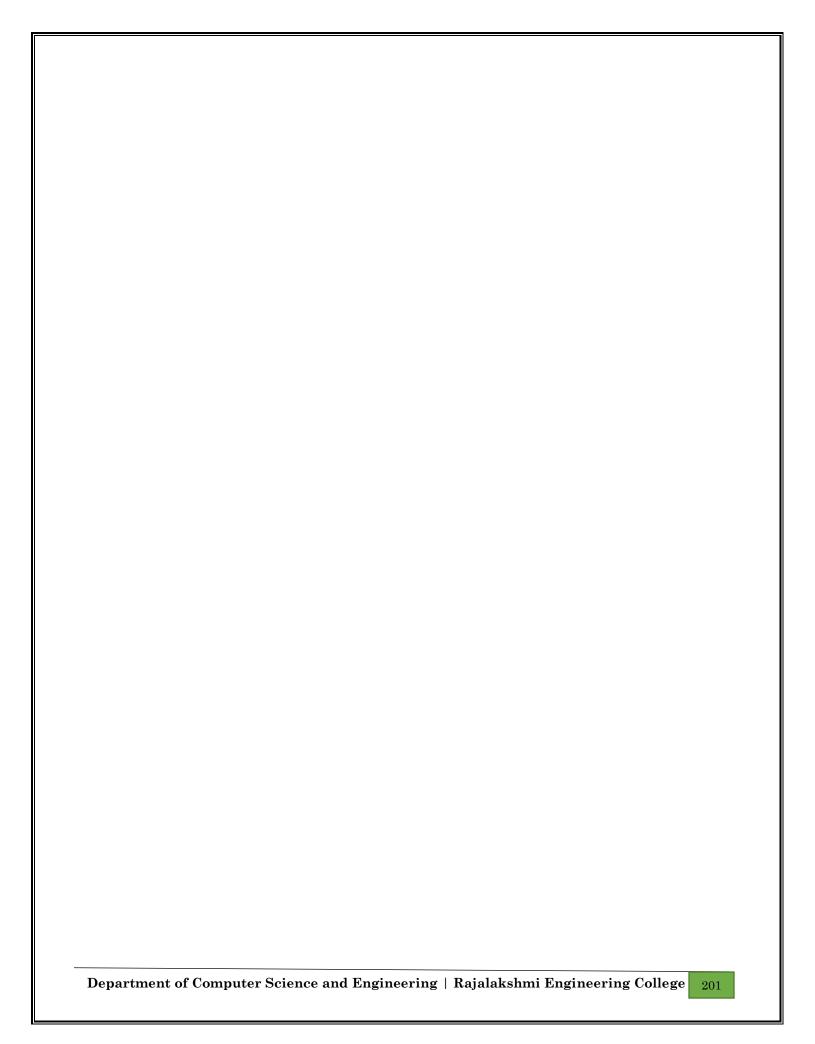
 $print("List\ is\ sorted\ in", count, "swaps.")$

b[i],b[i+1]=b[i+1],b[i]

 $print("First\ Element:",b[0])$

count+=1

print("Last Element:",b[-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

8 9 10 2 6

Sample Output

106

or caumpic.		
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]
def find_peak(arr):
  peaks = []
  n = len(arr)
  if n == 1:
     return arr[0]
  for i in range(n):
     if i == 0 and arr[i] \ge arr[i+1]:
       peaks.append(arr[i])
     elif i == n-1 and arr[i] >= arr[i-1]:
       peaks.append(arr[i])
     elif arr[i] \ge arr[i-1] and arr[i] \ge arr[i+1]:
        peaks.append(arr[i])
  return peaks
n = int(input())
arr = list(map(int, input().split()))
```

	peak_elements = find_peak(arr)
-	peak_elements = miu_peak(arr)
]	print(*peak_elements)

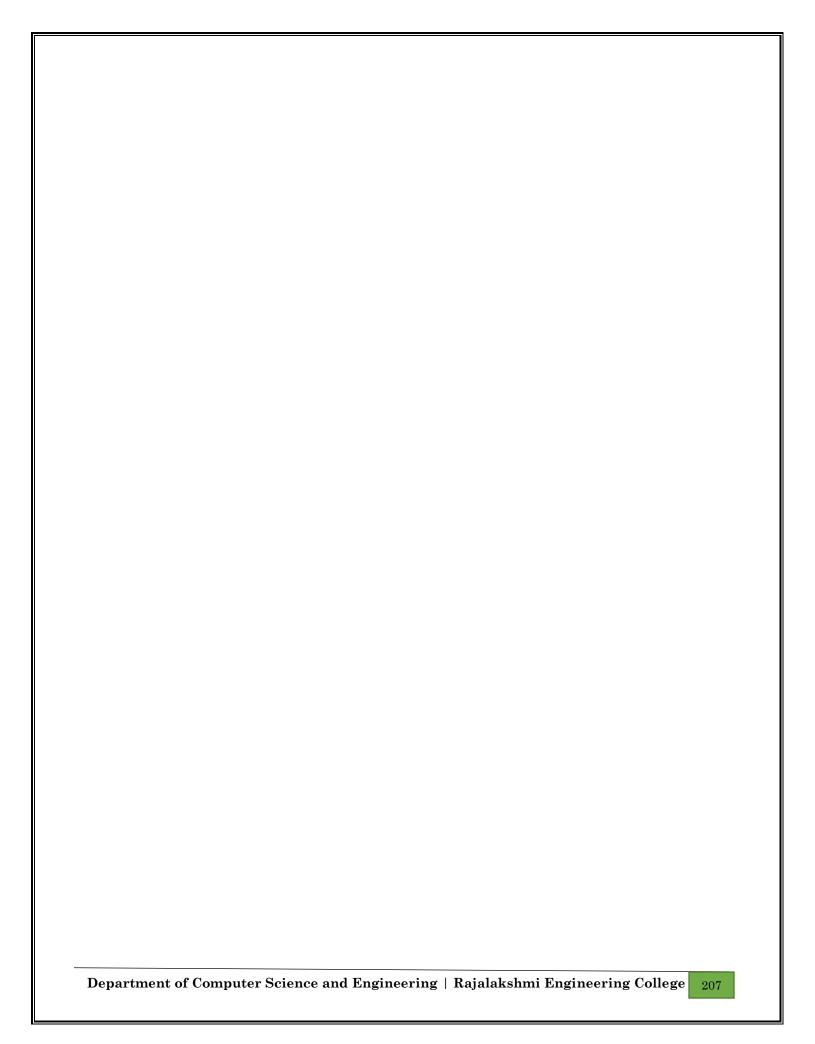
Input	Result
1 2 3 5 8 6	False
3 5 9 45 42 42	True

Ex. No.	:	10.4	Date:	
Register N	o.:		Name:	

Binary Search

Write a Python program for binary search.

```
def binary_search(arr, x):
  left = 0
  right = len(arr) - 1
  while left <= right:
     mid = left + (right - left) // 2
     if arr[mid] == x:
        return True
     elif arr[mid] < x:
        left = mid + 1
     else:
        right = mid - 1
  return False
def main():
  arr = list(map(int, input().strip().split(',')))
  x = int(input().strip())
  result = binary_search(sorted(arr), x)
  print(result)
main()
```



Input:

1 68 79 4 90 68 1 4 5

output:

1 2

4 2

5 1

682

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:

```
1<=n, arr[i]<=100
input_numbers = input().strip().split()
numbers = [int(x) for x in input_numbers]
frequency = {}
for number in numbers:
    if number in frequency:
        frequency[number] += 1
    else:
        frequency[number] = 1
sorted_numbers = sorted(frequency.keys())
for number in sorted_numbers:
    print(number, frequency[number])</pre>
```