# 10 - Searching & Sorting

For example:

| 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.

def mergeSort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left\_half = arr[:mid]

right\_half = arr[mid:]

mergeSort(left\_half)

mergeSort(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

return arr

def main():

n = int(input())

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

sorted\_arr = mergeSort(arr)

for i in sorted\_arr:

print(i,end=" ")

main()

Input Format

The first line contains an integer,n , the size of the [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068) 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](http://118.185.187.137/moodle/mod/resource/view.php?id=1068) 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](http://118.185.187.137/moodle/mod/resource/view.php?id=1068).

3.      Last Element: lastElement, the last element in the sorted [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068).

Sample Input 0

3

1 2 3

Sample Output 0

[List](http://118.185.187.137/moodle/mod/resource/view.php?id=1068) 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 |

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.      [List](http://118.185.187.137/moodle/mod/resource/view.php?id=1068) 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](http://118.185.187.137/moodle/mod/resource/view.php?id=1068).

3.      Last Element: lastElement, the last element in the sorted [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068).

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):

check = True

swap = 0

while check:

check = False

for j in range(1,len(arr)):

if arr[j-1]> arr[j]:

arr[j-1], arr[j] = arr[j],arr[j-1]

swap +=1

check = True

return swap

def main():

n = int(input())

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

swap = bubblesort(arr)

print(f"List is sorted in {swap} swaps.")

print(f"First Element:{arr[0]}")

print(f"Last Element:{arr[-1]}")

print()

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

10 6

For example:

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

Ex. No. : 10.3 Date:

Register No.: Name:

Peak Element

Given an [list](http://118.185.187.137/moodle/mod/resource/view.php?id=1068), 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]

def Greater(arr):

if arr[0] > arr[1]:

print(arr[0],end=" ")

for i in range(1,len(arr)-1):

if arr[i]> arr[i-1] and arr[i]> arr[i+1]:

print(arr[i],end=" ")

if arr[len(arr)-1 ] > arr[len(arr)-2]:

print(arr[len(arr)-1],end=" ")

def main():

n = int(input())

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

Greater(arr)

main()

For example:

| Input | Result |
| --- | --- |
| 1 2 3 5 8  6 | 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.

def binarySearch(arr, key):

s = 0

l = len(arr) - 1

while s <= l:

m = round((s + l) / 2)

if arr[m] == key:

return True

elif key < arr[m]:

l = m - 1

else:

s = m + 1

return False

def main():

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

n = int(input())

arr = sorted(arr)

if binarySearch(arr, n):

print("True")

else:

print("False")

main()

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 |

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

def Freq(arr,n):

temp = [0]\*n

arr = sorted(arr)

myset = set(arr)

for i in myset:

temp[i] = arr.count(i)

arr = sorted(list(myset))

for i in arr:

print(i,temp[i])

def main():

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

Freq(arr,100)

main()