**Sorting**

**Big Sorting**

import math

import os

import random

import re

import sys

def bigSorting(unsorted):

    return sorted(unsorted, key=lambda x: (len(x), x))

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    n = int(input())

    unsorted = []

    for \_ in range(n):

        unsorted\_item = input()

        unsorted.append(unsorted\_item)

    result = bigSorting(unsorted)

    fptr.write('\n'.join(result))

    fptr.write('\n')

    fptr.close()

# Intro to Tutorial Challenges

import math

import os

import random

import re

import sys

def introTutorial(V, arr):

    return arr.index(V)

    if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    V = int(input())

    n = int(input())

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

    result = introTutorial(V, arr)

    fptr.write(str(result) + '\n')

    fptr.close()

# Insertion Sort - Part 1

import math

import os

import random

import re

import sys

def insertionSort1(n, arr):

    i = n-1

    v = arr[i]

    while(i>0 and v<arr[i-1]):

        arr[i] = arr[i-1]

        print(\*arr)

        i-=1

    arr[i] = v

    print(\*arr)

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

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

    insertionSort1(n, arr)

# Insertion Sort - Part 2

import math

import os

import random

import re

import sys

def insertionSort2(n, arr):

    for i in range(n):

        if(i==0):

            continue

        for j in range(0,i):

            if (arr[j]>arr[i]):

                temp=arr[i]

                arr[i]=arr[j]

                arr[j]=temp

            else:

                continue

        print(\*arr)

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

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

    insertionSort2(n, arr)

# Correctness and the Loop Invariant

def insertion\_sort(l):

    for i in range(1, len(l)):

        j = i-1

        key = l[i]

        while (j > 0) and (l[j] > key):

           l[j+1] = l[j]

           j -= 1

        l[j+1] = key

    print(' '.join([str(n) for n in sorted(l)]))

m = int(input().strip())

ar = [int(i) for i in input().strip().split()]

insertion\_sort(ar)

# Running Time of Algorithms

import math

import os

import random

import re

import sys

def runningTime(arr):

    n=0

    if arr==sorted(arr):

        return (0)

    else:

        for i in range(1,len(arr)):

            p=arr[i]

            for j in range(i):

                if arr[j]>p:

                    (arr[i],arr[j])=(arr[j],arr[i])

                    n+=1

        return n

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    n = int(input())

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

    result = runningTime(arr)

    fptr.write(str(result) + '\n')

    fptr.close()

# Quicksort 1 - Partition

n= int(input())

s= str(input()).split()

l=list()

for i in range(n):

    l.append(int(s[i]))

m=list()

j=list()

for i in range(1,n):

    if l[i] > l[0]:

        m.append(l[i])

    elif l[i]<l[0]:

        j.append(l[i])

h=list()

j.sort(reverse = True)

for i in j:

    h.append(str(i))

h.append(str(l[0]))

m.sort()

for i in m:

    h.append(str(i))

b=" ".join(h)

print(b)

# Counting Sort 1

n = int(input())

l = [0]\*100

for i in map(int,input().split()):

    l[i]+=1

print(\*l)

# Counting Sort 2

n = int(input())

cnt = [0] \* 100

for i in map(int, input().split()):

    cnt[i] += 1

print ("".join((str(i) + " ") \* x for i, x in enumerate(cnt)))

# Find the Median

import math

import os

import random

import re

import sys

def findMedian(arr):

    arr.sort()

    if len(arr)%2 == 0:

        print(arr[len(arr)/2]+arr[len((arr/2)+1)])

    else:

        print(arr[int(len(arr)/2)])

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

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

    result = findMedian(arr)