**10 - Searching & Sorting**

**Ex:10.1 Merge Sort**

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

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  6 5 4 3 8 | 3 4 5 6 8 |

PROGRAM:

def merge(arr, l, m, r):  
n1 = m - l + 1  
n2 = r - m  
  
# create temp arrays  
L = [0] \* (n1)  
R = [0] \* (n2)  
  
# Copy data to temp arrays L[] and R[]  
for i in range(0, n1):  
L[i] = arr[l + i]  
  
for j in range(0, n2):  
R[j] = arr[m + 1 + j]  
  
# Merge the temp arrays back into arr[l..r]  
i = 0 # Initial index of first subarray  
j = 0 # Initial index of second subarray  
k = l # Initial index of merged subarray  
  
while i < n1 and j < n2:  
if L[i] <= R[j]:  
arr[k] = L[i]  
i += 1  
else:  
arr[k] = R[j]  
j += 1  
k += 1  
  
# Copy the remaining elements of L[], if there  
# are any  
while i < n1:  
arr[k] = L[i]  
i += 1  
k += 1  
  
# Copy the remaining elements of R[], if there  
# are any  
while j < n2:  
arr[k] = R[j]  
j += 1  
k += 1  
  
# l is for left index and r is right index of the  
# sub-array of arr to be sorted  
  
  
def mergeSort(arr, l, r):  
if l < r:  
  
# Same as (l+r)//2, but avoids overflow for  
# large l and h  
m = l+(r-l)//2  
  
# Sort first and second halves  
mergeSort(arr, l, m)  
mergeSort(arr, m+1, r)  
merge(arr, l, m, r)  
  
  
# Driver code to test above  
n=int(input())  
l=list(map(int,input().split()))  
mergeSort(l,0,len(l)-1)  
  
print(\*l)

**Ex:10.2 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

**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 |

PROGRAM:

n = int(input())

l = input().split()

l = [int(i) for i in l]

sw = 0

for i in range(n):

for j in range(n-i-1):

if(l[j]>l[j+1]):

l[j],l[j+1] = l[j+1],l[j]

sw+=1

print("List is sorted in %d swaps."%sw)

print("First Element:",l[0])

print("Last Element:",l[n-1]) **Ex:10.3 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]

**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 |

PROGRAM:

n = int(input())

ip = input().split()

ans = []

ip = [int(i) for i in ip]

for i in range(len(ip)):

if(i==0):

if(ip[i]>ip[i+1]):

ans.append(ip[i])

elif(i==len(ip)-1):

if(ip[i]>ip[i-1]):

ans.append(ip[i])

else:

if(ip[i]>ip[i+1] and ip[i]>ip[i-1]):

ans.append(ip[i])

for i in ans:

print(i,end = ' ')

**Ex:10.4 Binary Search**

Write a Python program for binary search.

**For example:**

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

PROGRAM:

s = input().split(',')

s = [int(i) for i in s]

n = int(input())

f = 0

mid = s[len(s)-1]

low = s[0]

high = s[len(s)-1]

if(mid==n):

f=1

if(f==0):

while(low!=mid and high!=mid):

if(mid<n):

low = s[mid+1]

mid = [(low+high)//2]

elif(mid>n):

high = s[mid-1]

mid = [(low+high)//2]

else:

f = 1

break

if(f==1):

print(True)

else:

print(False)

**Ex:10.5 Frequency of Elements**

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

**Constraints:**

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

**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 |

PROGRAM:

s = input().split()

s = [int(i) for i in s]

l = set(s)

l = list(l)

l.sort()

for i in l:

print(i,s.count(i))