Ex. No. : 10.1 Date: 7/6/2024

Register No.: 230401023 Name: CAROLINE

Merge Sort

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

```
def merge-sort (arr):
    if len (arr)>1:
        mid= len(arr) //2
          L= arr [ : m:d]
          R=arr [ m:d:]
        merge-sort (L)
        merge-sort (R)
        i = j = k =0
      while i <len (L) andj < len (R):
        if L[i] <R [j]:
          arr [R] = L[i]
            i += 1

        else:
          arr [k] = R[j]
            j +=1
            k+=1
      while i < len (L):
          arr[R] = L[i]
            i+=1
            R+=1
      whilj < len[R]:
          arr [R] = R[j]
          j +1   R+1
```

Ex. No. : 10.2 Date: 7/6/2024

Register No.: 230401023 Name: CAROLINE

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 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 bubble_sort (arr):
        num_swap =0
        n=len(arr)
    for i in range (n):
          swapped = false
    for j in range ( 0, n-j-1):
        if arr [j] > arr [j+1]:
              arr[j] ,arr [j+1] = arr[j+1], arr [i]
              num_swaps +=1
              swapped = True
    if not swapped:
              break
        return num_swaps
    n=int(input())
arr = list ( map (int, input() .split()))
Print ("List is sorted in", num_swaps," swaps")
```

Ex. No. : 10.3 Date: 7/6/2024

Register No.: 230401023 Name: CAROLINE

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] <= 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]

```
a= int( input ())
b= input ().split()
x= list( map(int, b))
y = []
for i in range(len(x)):
    if (i==0 or x[i]>= x[i-1] and i==len(x)-1 or
            x[i]>= x[i+1]):
            y.append (x[i])
    for i in range (len(y'):
        print( y[i], end=" ").
```

No. : 10.4 Date: 7/6/2024

gister No.: 230401023 Name: CAROLINE

Binary Search

ite a Python program for binary search.

```
def binary_search(arr, x):
    arr.sort()
    left, right=0, len(arr)-1
    while left <= right:
        mid = (left + right)//2
        if arr [mid] ==x
            return True
        elif are [mid] <x:
            left = mid +1
        else:
            right =mid -1

    return False
numbers =list (map(int, input().split (',')))
 target= int (input())
 result = binary_search( numbers, target)
 Print (result)
```

x. No. : 10.5 Date: 7/6/2024

egister No.: 230401023 Name: CAROLINE

Frequency of Elements

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

onstraints:

<=n, arr[i]<=100

```
numbu = list (map (int input()) .split ()))
frequency= { }
for num in frequency:
    if   num in frequency:
      frequency [num]+ = 1
  else:
      frequency [num] = 1

sorted_ fouquency = sorted (frequency .items())
 for num , freq in sorted_frequency:
     print (num, freq )
```

AArun