vl dharshini
231501038
aiml a

1) Write a Python program for binary search.

PROGRAM:

```
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 arr[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)
```

OUTPUT:

	Input	Expected	Got	
✔	1,2,3,5,8 6	False	False	✔
✔	3,5,9,45,42 42	True	True	✔
✔	52,45,89,43,11 11	True	True	✔

Passed all tests! ✔

2) 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]

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
```

PROGRAM:

```
def find_peaks(arr):
    n = len(arr)
    peaks = []
    if n > 0 and (n == 1 or arr[0] >= arr[1]):
        peaks.append(arr[0])
    for i in range(1, n-1):
        if arr[i] >= arr[i-1] and arr[i] >= arr[i+1]:
            peaks.append(arr[i])
    if n > 1 and arr[n-1] >= arr[n-2]:
        peaks.append(arr[n-1])
    return peaks
if __name__ == "__main__":
    n = int(input())
    arr = list(map(int, input().split()))
```

```
peaks = find_peaks(arr)
print(" ".join(map(str, peaks)))
```

OUTPUT:

	Input	Expected	Got	
✔	7 15 7 10 8 9 4 6	15 10 9 6	15 10 9 6	✔
✔	4 12 3 6 8	12 8	12 8	✔

Passed all tests! ✔

3) Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. You read an list of numbers. You need to arrange the elements in ascending order and print the result. The sorting should be done using bubble sort.

Input Format: The first line reads the number of elements in the array. The second line reads the array elements one by one.

Output Format: The output should be a sorted list.

PROGRAM:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]
        merge_sort(left_half)
        merge_sort(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
n = int(input( ))
arr = list(map(int, input().split()))
merge_sort(arr)
print(' '.join(map(str, arr)))
```

OUTPUT:

	Input	Expected	Got	
✔	6 3 4 8 7 1 2	1 2 3 4 7 8	1 2 3 4 7 8	✔
✔	6 9 18 1 3 4 6	1 3 4 6 9 18	1 3 4 6 9 18	✔
✔	5 4 5 2 3 1	1 2 3 4 5	1 2 3 4 5	✔

Passed all tests! ✔

4) 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

PROGRAM:

```
def find_frequencies(arr):
    frequency_dict = {}
    for number in arr:
        if number in frequency_dict:
            frequency_dict[number] += 1
        else:
            frequency_dict[number] = 1
```

```
    sorted_frequency = sorted(frequency_dict.items())
    return sorted_frequency
if __name__ == "__main__":
    arr = list(map(int, input().split()))
    frequencies = find_frequencies(arr)
    for number, frequency in frequencies:
        print(number,frequency)
```

OUTPUT:

	Input	Expected	Got	
✔	4 3 5 3 4 5	3 2 4 2 5 2	3 2 4 2 5 2	✔
✔	12 4 4 4 2 3 5	2 1 3 1 4 3 5 1 12 1	2 1 3 1 4 3 5 1 12 1	✔
✔	5 4 5 4 6 5 7 3	3 1 4 2 5 3 6 1 7 1	3 1 4 2 5 3 6 1 7 1	✔

Passed all tests! ✔

5) 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
```

Input Format

The first line contains an integer,n , the size of the list 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. 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.

Sample Input 0

3

1 2 3

Sample Output 0

List is sorted in 0 swaps.

First Element: 1

Last Element: 3

PROGRAM:

```
def bubbleSort(arr):
    n = len(arr)
    numSwaps = 0

    for i in range(n):
        swapped = False
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

```
            swapped = True
            numSwaps += 1

        # If no swaps occurred in this pass, the list is already sorted
        if not swapped:
            break

    return numSwaps

# Input
n = int(input())
arr = list(map(int, input().split()))

# Sort the array and get the number of swaps
numSwaps = bubbleSort(arr)

# Print the required information
print(f"List is sorted in {numSwaps} swaps.")
print(f"First Element: {arr[0]}")
print(f"Last Element: {arr[-1]}")
```

OUTPUT:

	Input	Expected	Got	
✔	3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3	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	List is sorted in 4 swaps. First Element: 1 Last Element: 9	✔

Passed all tests! ✔

WEEK 10