161. Implement the Quick Sort algorithm in a programming language of your choice and test it on the array 19,72,35,46,58,91,22,31. Choose the middle element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display the array after each recursive call until the entire array is sorted. Execute your code and show the sorted array.

Input : N= 8, a[] = {19,72,35,46,58,91,22,31}
Output : 19,22,31,35,46,58,72,91
Test Cases :
Input : N= 8, a[] = {31,23,35,27,11,21,15,28}
Output : 11,15,21,23,27,28,31,35
Test Cases :
Input : N= 10, a[] = {22,34,25,36,43,67, 52,13,65,17}
Output : 13,17,22,25,34,36,43,52,65,67

PROGRAM :-

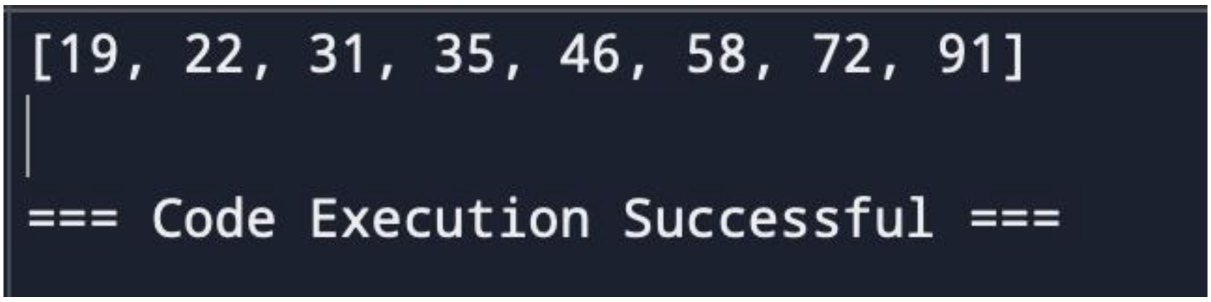
```
def quick_sort(arr):
   if len(arr) <= 1:
      return arr
   else:
      pivot = arr[len(arr) // 2]
      left = [x for x in arr if x < pivot]
      middle = [x for x in arr if x == pivot]
      right = [x for x in arr if x > pivot]
      return quick_sort(left) + middle + quick_sort(right)

# Input array
arr = [19, 72, 35, 46, 58, 91, 22, 31]

# Sorting the array using Quick Sort
sorted_array = quick_sort(arr)

# Displaying the sorted array
print(sorted_array)
```

OUTPUT:-



```
[19, 22, 31, 35, 46, 58, 72, 91]

=== Code Execution Successful ===
```

TIME COMPLEXITY:- O(N log N)