

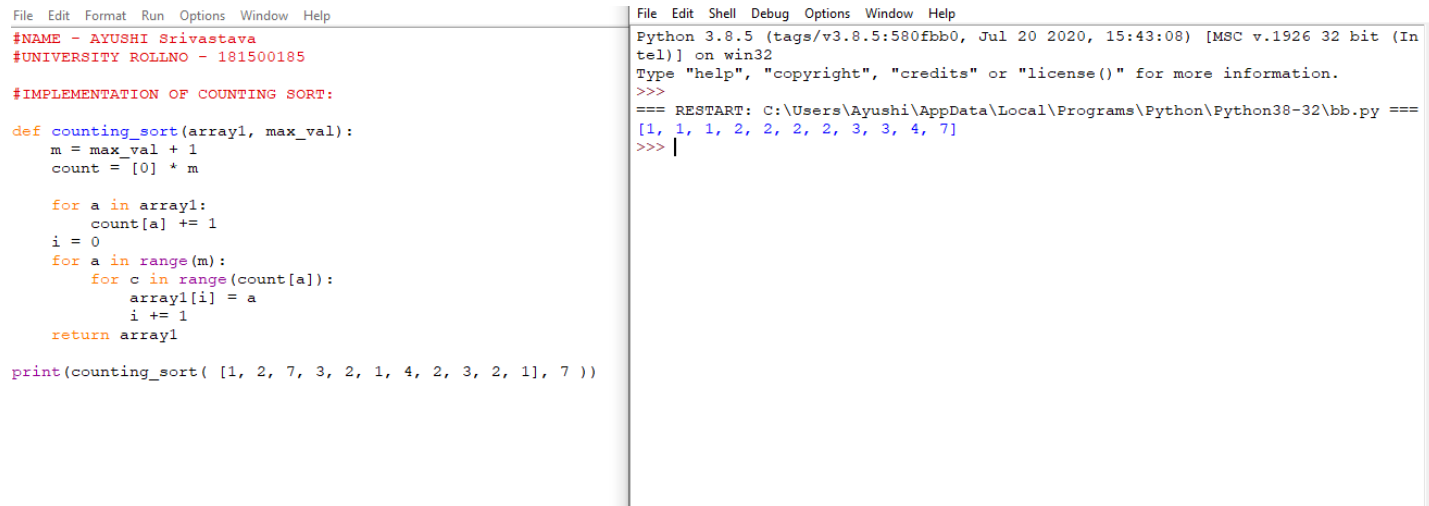
CODE:

```
def counting_sort(array1, max_val):  
    m = max_val + 1  
    count = [0] * m  
  
    for a in array1:  
        count[a] += 1  
    i = 0  
    for a in range(m):  
        for c in range(count[a]):  
            array1[i] = a  
            i += 1  
    return array1  
  
print(counting_sort( [1, 2, 7, 3, 2, 1, 4, 2, 3, 2, 1], 7 ))
```

Inputs: [1,2,7,3,2,1,4,2,3,2,1],7))

Output:[1,1,1,2,2,2,3,3,4,7]

● Screen Shot Of O/P:-



The screenshot displays a Python IDE with two windows. The left window shows the implementation of a counting sort function, and the right window shows the execution output.

```
File Edit Format Run Options Window Help
#NAME - AYUSHI Srivastava
#UNIVERSITY ROLLNO - 181500185

#IMPLEMENTATION OF COUNTING SORT:

def counting_sort(array1, max_val):
    m = max_val + 1
    count = [0] * m

    for a in array1:
        count[a] += 1
    i = 0
    for a in range(m):
        for c in range(count[a]):
            array1[i] = a
            i += 1
    return array1

print(counting_sort( [1, 2, 7, 3, 2, 1, 4, 2, 3, 2, 1], 7 ))
```

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\Ayushi\AppData\Local\Programs\Python\Python38-32\bb.py ===
[1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 7]
>>>
```

Time Complexity of Counting Sort Algorithm:

- Best Case $O(n+k)$
- Average Case $O(n+k)$
- Worst Case $O(n+k)$, where n is the size of the input array and k means the * values range from 0 to k .