Counting Sort 2

Often, when a list is sorted, the elements being sorted are just keys to other values. For example, if you are sorting files by their size, the sizes need to stay connected to their respective files. You cannot just take the size numbers and output them in order, you need to output all the required file information.

The *counting sort* is used if you just need to sort a list of integers. Rather than using a comparison, you create an integer array whose index range covers the entire range of values in your array to sort. Each time a value occurs in the original array, you increment the counter at that index. At the end, run through your counting array, printing the value of each non-zero valued index that number of times.

For example, consider an array $arr = [1, 1, 3, 2, 1]$. All of the values are in the range $[0 \ldots 3]$, so create an array of zeroes, $result = [0, 0, 0, 0]$. The results of each iteration follow:

```
i       arr[i]  result
0       1       [0, 1, 0, 0]
1       1       [0, 2, 0, 0]
2       3       [0, 2, 0, 1]
3       2       [0, 2, 1, 1]
4       1       [0, 3, 1, 1]
```

Now we can print the sorted array: $sorted = [1, 1, 1, 2, 3]$.

Challenge

Given an unsorted list of integers, use the counting sort method to sort the list and then print the sorted list.

Hint: You can use your previous code that counted the items to print out the actual values in order.

Function Description

Complete the *countingSort* function in the editor below. It should return the original array, sorted ascending, as an array of integers.

countingSort has the following parameter(s):

- *arr*: an array of integers

Input Format

The first line contains an integer n, the length of arr. The next line contains space-separated integers $arr[i]$ where $0 \leq i < n$.

Constraints

$0 \leq n \leq 1000000$
$0 \leq arr[i] < 100$

Output Format

Print the sorted list as a single line of space-separated integers.

Sample Input

```
100
63 25 73 1 98 73 56 84 86 57 16 83 8 25 81 56 9 53 98 67 99 12 83 89 80 91 39 86 76 85 74 39 25 90 59 10 94
32 44 3 89 30 27 79 46 96 27 32 18 21 92 69 81 40 40 34 68 78 24 87 42 69 23 41 78 22 6 90 99 89 50 30 20 1
43 3 70 95 33 46 44 9 69 48 33 60 65 16 82 67 61 32 21 79 75 75 13 87 70 33
```

Sample Output

```
1 1 3 3 6 8 9 9 10 12 13 16 16 18 20 21 21 22 23 24 25 25 25 27 27 30 30 32 32 32 33 33
33 34 39 39 40 40 41 42 43 44 44 46 46 48 50 53 56 56 57 59 60 61 63 65 67 67 68 69 69 69
70 70 73 73 74 75 75 76 78 78 79 79 80 81 81 82 83 83 84 85 86 86 87 87 89 89 89 90 90 91
92 94 95 96 98 98 99 99
```

Explanation

Once our counting array has been filled, loop from index 0 to the end, printing each i value $arr[i]$ times.