Counting Sort:

Counting Sort is a non-comparison-based sorting algorithm that works well when there is limited range of input values. It is particularly efficient when the range of input values is small compared to the number of elements to be sorted. The basic idea behind Counting Sort is to count the frequency of each distinct element in the input array and use that information to place the elements in their correct sorted positions.

Algorithm:

1. Declare an auxiliary array countArray[] of size max(inputArray[])+1 and initialize it with 0.
2. Traverse array inputArray[] and map each element of inputArray[] as an index of countArray[] array, i.e., execute countArray[inputArray[i]]++ for 0 <= i < N.
3. Calculate the prefix sum at every index of array inputArray[].
4. Create an array outputArray[] of size N.
5. Traverse array inputArray[] from end and update outputArray[countArray[inputArray[i]] – 1] = inputArray[i]. Also, update countArray[inputArray[i]] = countArray[inputArray[i]].

Code:

```
#include <bits/stdc++.h>
using namespace std;

vector<int> countSort(vector<int>& inputArray)
{

	int N = inputArray.size();

	// Finding the maximum element of array inputArray[].
	int M = 0;

	for (int i = 0; i < N; i++)
		M = max(M, inputArray[i]);

	// Initializing countArray[] with 0
	vector<int> countArray(M + 1, 0);

	// Mapping each element of inputArray[] as an index
```

```
        // of countArray[] array

        for (int i = 0; i < N; i++)
                countArray[inputArray[i]]++;

        // Calculating prefix sum at every index
        // of array countArray[]
        for (int i = 1; i <= M; i++)
                countArray[i] += countArray[i - 1];

        // Creating outputArray[] from countArray[] array
        vector<int> outputArray(N);

        for (int i = N - 1; i >= 0; i--)

        {
                outputArray[countArray[inputArray[i]] - 1]
                        = inputArray[i];

                countArray[inputArray[i]]--;
        }

        return outputArray;
}

// Driver code
int main()

{

        // Input array
        vector<int> inputArray = { 4, 3, 12, 1, 5, 5, 3, 9 };

        // Output array
        vector<int> outputArray = countSort(inputArray);

        for (int i = 0; i < inputArray.size(); i++)
                cout << outputArray[i] << " ";

        return 0; }
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    SEAR

WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In
derr=Microsoft-MIEngine-Error-xi3qk4ll.b10' '--pid=Micro
ysys2\ucrt64\bin\gdb.exe' '--interpreter=mi'
1 3 3 4 5 5 9 12
PS D:\Mirror\ICT\3rd YEAR\SEM 5\Design and Analysis of A
```

Space complexity: _____

Justification:_____
_____

Time complexity:

Best case time complexity: _____

Justification:_____
_____

Worst case time complexity: _____

Justification:_____