Merge Sort in C++

```
#include <iostream>
#include <vector>
using namespace std;

class MergeSort {
public:
    void merge(vector<int>& arr, int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;

        // Create temporary arrays
        vector<int> L(n1), R(n2);

        // Copy data to temporary arrays L[] and R[]
        for (int i = 0; i < n1; i++)
            L[i] = arr[l + i];
        for (int j = 0; j < n2; j++)
            R[j] = arr[m + 1 + j];

        // Merge the temporary arrays back into arr[l..r]
        int i = 0; // Initial index of first subarray
        int j = 0; // Initial index of second subarray
        int k = l; // Initial index of merged subarray

        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            } else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        // Copy the remaining elements of L[], if any
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }

        // Copy the remaining elements of R[], if any
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

    void mergeSort(vector<int>& arr, int l, int r) {
        if (l >= r) {
            return; // Base case: array size is 0 or 1
        }
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);     // Sort first half
        mergeSort(arr, m + 1, r); // Sort second half
        merge(arr, l, m, r);      // Merge sorted halves
    }
};
```

Let's walk through a **dry run** of your **Merge Sort implementation** with the input:

arr = {12, 11, 13, 5, 6, 7}

Step-by-step Breakdown:

We'll visualize the recursive division and merging process.

Recursive Division (mergeSort)

Level	Call	Subarray
1	mergeSort(arr, 0, 5)	[12, 11, 13, 5, 6, 7]
2	mergeSort(arr, 0, 2)	[12, 11, 13]
3	mergeSort(arr, 0, 1)	[12, 11]
4	mergeSort(arr, 0, 0)	[12]
4	mergeSort(arr, 1, 1)	[11]
3	merge(arr, 0, 0, 1)	merge [12] and [11] ⇒ [11, 12]
3	mergeSort(arr, 2, 2)	[13]
2	merge(arr, 0, 1, 2)	merge [11, 12] and [13] ⇒ [11, 12, 13]
2	mergeSort(arr, 3, 5)	[5, 6, 7]
3	mergeSort(arr, 3, 4)	[5, 6]
4	mergeSort(arr, 3, 3)	[5]
4	mergeSort(arr, 4, 4)	[6]
3	merge(arr, 3, 3, 4)	merge [5] and [6] ⇒ [5, 6]
3	mergeSort(arr, 5, 5)	[7]
2	merge(arr, 3, 4, 5)	merge [5, 6] and [7] ⇒ [5, 6, 7]

```
int main() {
    MergeSort solution;

    // Hardcoded input array
    vector<int> arr = {12, 11, 13, 5, 6, 7};
    int n = arr.size();

    cout << "Given Array:" << endl;
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    solution.mergeSort(arr, 0, n - 1);

    cout << "\nSorted array:" << endl;
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

1	merge(arr, 0, 2, 5)	merge [11, 12, 13] and [5, 6, 7] ⇒ [5, 6, 7, 11, 12, 13]

✅ **Final Sorted Array:**

[5, 6, 7, 11, 12, 13]

📓 **Visual of Merges**

Initial: [12, 11, 13, 5, 6, 7]
Split1: [12, 11, 13] | [5, 6, 7]
Split2: [12, 11] [13] | [5, 6] [7]
Merge1: [11, 12] + [13] = [11, 12, 13]
Merge2: [5, 6] + [7] = [5, 6, 7]
Final Merge: [11, 12, 13] + [5, 6, 7] = [5, 6, 7, 11, 12, 13]

```
Given Array:
12 11 13 5 6 7

Sorted array:
5 6 7 11 12 13
```