MergeSort in C++

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

void merge(int arr[], int l, int m, int r) {
    int n1=m-l+1;
    int n2=r-m;
    int left[n1];
    int right[n2];

    for(int i=0;i<n1;i++){
        left[i]=arr[l+i];
    }

    for(int j=0;j<n2;j++){
        right[j]=arr[m+1+j];
    }
    int i = 0, j = 0, k = l;

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

     while (i < n1) {
         arr[k]=left[i];
         i++;
         k++;
     }

    while(j<n2){
        arr[k]=right[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r) {
    if (l >= r) {
        return;
    }
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);
    merge(arr, l, m, r);
}

int main() {
    /* Enter your code here. Read input from STDIN.
Print output to STDOUT */

    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
```

Example Input:
n = 6
arr = {38, 27, 43, 3, 9, 82}

🧮 Merge Sort Recursive Dry Run (Call Stack Overview)

Call Level	Function Call	Action	Array State
1	mergeSort(0, 5)	Split at 2	
2	mergeSort(0, 2)	Split at 1	
3	mergeSort(0, 1)	Split at 0	
4	mergeSort(0, 0)	Base case	[38]
4	mergeSort(1, 1)	Base case	[27]
3	merge(0, 0, 1)	Merge [38] & [27] → [27, 38]	[27, 38, 43, 3, 9, 82]
2	mergeSort(2, 2)	Base case	[43]
2	merge(0, 1, 2)	Merge [27, 38] & [43]	[27, 38, 43, 3, 9, 82]
1	mergeSort(3, 5)	Split at 4	
2	mergeSort(3, 4)	Split at 3	
3	mergeSort(3, 3)	Base case	[3]
3	mergeSort(4, 4)	Base case	[9]
2	merge(3, 3, 4)	Merge [3] & [9] → [3, 9]	[27, 38, 43, 3, 9, 82]
1	mergeSort(5, 5)	Base case	[82]
1	merge(3, 4, 5)	Merge [3, 9] & [82]	[27, 38, 43, 3, 9, 82]
0	merge(0, 2, 5)	Merge [27, 38, 43] & [3, 9, 82] →	

```
        cin >> arr[i];
    }

    mergeSort(arr,0,n-1);

    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
    return 0;
}
```

		[3, 9, 27, 38, 43, 82]	

✅ Final Output:
3 9 27 38 43 82

```
3 9 27 38 43 82
```