MergeSort in C++

```
// C++ Implementation of Merge Sort
#include <iostream>
using namespace std;

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

   // Create temp arrays
   int L[n1], R[n2];

   // Copy data to temp arrays
   for (int i = 0; i < n1; i++)
      L[i] = arr[left + i];
   for (int j = 0; j < n2; j++)
      R[j] = arr[mid + 1 + j];


   // Merge temp arrays back into arr[left..right]
   int i = 0;    // Initial index of first subarray
   int j = 0;    // Initial index of second subarray
   int k = left; // 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 remaining elements of L[] if any
   while (i < n1) {
      arr[k] = L[i];
      i++;
      k++;
   }

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

void mergeSort(int arr[], int left, int right) {
```

```
    if (left < right) {
        int mid = left + (right - left) / 2;


        // Sort first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);

    }
}

int main() {
    int n;
    cin >> n;

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


    mergeSort(arr, 0, n-1);

    cout << "\nSorted array: ";
    for(int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Merge Sort in Java

```
// Java Implementation of Merge Sort
import java.util.Scanner;

class Main {
    public static void merge(int arr[], int left, int mid, int right) {
        int n1 = mid - left + 1;
        int n2 = right - mid;

        // Create temp arrays
        int L[] = new int[n1];
        int R[] = new int[n2];

        // Copy data to temp arrays
        for (int i = 0; i < n1; i++)
```

```
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    // Merge temp arrays
    int i = 0, j = 0; // Initial indexes of first and second subarrays
    int k = left;     // 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 remaining elements of L[] if any
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

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

public static void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;


        // Sort first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);


    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

```
        int n = scanner.nextInt();

        int[] arr = new int[n];
        for(int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }


        mergeSort(arr, 0, n-1);

        System.out.print("\nSorted array: ");
        for(int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();

        scanner.close();
    }
}
```