**MERGE SORT**

// Merge Sort function

function mergeSort(arr) {

    if (arr.length <= 1) {

        return arr;

    }

    const mid = Math.floor(arr.length / 2);

    const left = arr.slice(0, mid);

    const right = arr.slice(mid);

    return merge(mergeSort(left), mergeSort(right));

}

function merge(left, right) {

    let result = [];

    let leftIndex = 0;

    let rightIndex = 0;

    while (leftIndex < left.length && rightIndex < right.length) {

        if (left[leftIndex] < right[rightIndex]) {

            result.push(left[leftIndex]);

            leftIndex++;

        } else {

            result.push(right[rightIndex]);

            rightIndex++;

        }

    }

    return result.concat(left.slice(leftIndex), right.slice(rightIndex));

}

const array = [38, 27, 43, 3, 9, 82, 10];

const sortedArray = mergeSort(array);

console.log(sortedArray);

**OUTPUT**

**[3, 9, 10, 27,**

**38, 43, 82]**

**Input: [38, 27, 43, 3, 9, 82, 10]**

1. **First Split**:
   * Given array = [38, 27, 43, 3, 9, 82, 10]
   * Split into: left = [38, 27, 43], right = [3, 9, 82, 10]
2. **Recursively Split left**:
   * left = [38, 27, 43]
   * Split into: left = [38], right = [27, 43]
3. **Recursively Split right**:
   * right = [27, 43]
   * Split into: left = [27], right = [43]
   * Merging [27] and [43] gives [27, 43].
4. **Merge [38] and [27, 43]**:
   * [38] is compared with [27, 43].
   * Result: [27, 38, 43]
5. **Recursively Split right**:
   * right = [3, 9, 82, 10]
   * Split into: left = [3, 9], right = [82, 10]
6. **Recursively Split [3, 9]**:
   * left = [3], right = [9]
   * Merging gives: [3, 9]
7. **Recursively Split [82, 10]**:
   * left = [82], right = [10]
   * Merging gives: [10, 82]
8. **Merge [3, 9] and [10, 82]**:
   * Result: [3, 9, 10, 82]
9. **Merge [27, 38, 43] and [3, 9, 10, 82]**:
   * Final result: [3, 9, 10, 27, 38, 43, 82]

**Final Sorted Array:**

The sorted array is [3, 9, 10, 27, 38, 43, 82].