```
function merge(list1, list2) {
  if (list1.length < 1) {
    // List 1 is empty
    return list2;
  } else if (list2.length < 1) {
    // List 2 is empty
    return list1;
  } else {
    var first, rest, other;
    if (list1[0] < list2[0]) {
      // First element of list1 is smaller
      first = list1.slice(0, 1);
      rest = list1.slice(1, list1.length);
      other = list2;
    } else {
      // First element of list2 is smaller
      first = list2.slice(0, 1);
      rest = list2.slice(1, list2.length);
      other = list1;
    }
    return first.concat(merge(rest, other)); // Recursively merge the rest of the lists
  }
}

// Splits the list for merge sort
function mergeSort(array_list) {
  if (array_list.length <= 1) {
    return array_list;
  } else {
    return merge(
      mergeSort(array_list.slice(0, array_list.length / 2)),
      mergeSort(array_list.slice(array_list.length / 2, array_list.length))
    );
  }
}

module.exports = mergeSort
```

```
const mergeSort = require('./merge');

test('Merge sort with a empty list', () => {
  expect(mergeSort([])).toEqual([]);
});

test('Merge sort with a list of a single element', () => {
  expect(mergeSort([5])).toEqual([5]);
});

test('Merge sort with a list of two elements', () => {
  expect(mergeSort([5, 3])).toEqual([3, 5]);
});``

test('Merge sort with a list of sorted elements', () => {
  expect(mergeSort([2, 4, 6, 8, 10])).toEqual([2, 4, 6, 8, 10]);
});

test('Merge sort with a list of where the first half is sorted', () => {
  expect(mergeSort([2, 11, 15, 7, 3])).toEqual([2, 3, 7, 11, 15]);
});

test('Merge sort with a list of where the second half is sorted', () => {
  expect(mergeSort([7, 8, 5, 9, 13])).toEqual([5, 7, 8, 9, 13]);
});

test('Merge sort where all elements are unsorted', () => {
  expect(mergeSort([3, -1, 8, 2, 4])).toEqual([-1, 2, 3, 4, 8]);
});
```

Example Codebase w/ Jest

```
function merge(list1, list2) {
  if (list1.length < 1) {
    // List 1 is empty
    return list2;
  } else if (list2.length < 1) {
    // List 2 is empty
    return list1;
  } else {
    var first, rest, other;
    if (list1[0] < list2[0]) {
      // First element of list1 is smaller
      first = list1.slice(0, 1);
      rest = list1.slice(1, list1.length);
      other = list2;
    } else {
      // First element of list2 is smaller
      first = list2.slice(0, 1);
      rest = list2.slice(1, list2.length);
      other = list1;
    }
    return first.concat(merge(rest, other)); // Recursively merge the rest of the lists
  }
}

// Splits the list for merge sort
function mergeSort(array_list) {
  if (array_list.length <= 1) {
    return array_list;
  } else {
    return merge(
      mergeSort(array_list.slice(0, array_list.length / 2)),
      mergeSort(array_list.slice(array_list.length / 2, array_list.length))
    );
  }
}

module.exports = mergeSort
```

```
const mergeSort = require('./merge');

test('Merge sort with a empty list', () => {
  expect(mergeSort([])).toEqual([]);
});

test('Merge sort with a list of a single element', () => {
  expect(mergeSort([5])).toEqual([5]);
});

test('Merge sort with a list of two elements', () => {
  expect(mergeSort([5, 3])).toEqual([3, 5]);
});``

test('Merge sort with a list of sorted elements', () => {
  expect(mergeSort([2, 4, 6, 8, 10])).toEqual([2, 4, 6, 8, 10]);
});

test('Merge sort with a list of where the first half is sorted', () => {
  expect(mergeSort([2, 11, 15, 7, 3])).toEqual([2, 3, 7, 11, 15]);
});

test('Merge sort with a list of where the second half is sorted', () => {
  expect(mergeSort([7, 8, 5, 9, 13])).toEqual([5, 7, 8, 9, 13]);
});

test('Merge sort where all elements are unsorted', () => {
  expect(mergeSort([3, -1, 8, 2, 4])).toEqual([-1, 2, 3, 4, 8]);
});
```


module.exports = mergeSort
expect(mergeSort([3, -1, 8, 2, 4])).toEqual([-1, 2, 3, 4, 8]);
test('Merge sort where all elements are unsorted.', () => {