Our Solution(s) Run Code

Solution 1 Solution 2

```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class LinkedList {
  constructor(value) {
    this.value = value;
    this.next = null;
  }
}

// O(n + m) time | O(n + m) space - where n is the number of nodes in
// Linked List and m is the number of nodes in the second Linked List
function mergeLinkedLists(headOne, headTwo) {
  recursiveMerge(headOne, headTwo, null);
  return headOne.value < headTwo.value ? headOne : headTwo;
}

function recursiveMerge(p1, p2, p1Prev) {
  if (p1 === null) {
    p1Prev.next = p2;
    return;
  }
  if (p2 === null) return;

  if (p1.value < p2.value) {
    recursiveMerge(p1.next, p2, p1);
  } else {
    if (p1Prev !== null) p1Prev.next = p2;
    const newP2 = p2.next;
    p2.next = p1;
    recursiveMerge(p1, newP2, p2);
  }
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```
// This is an input class. Do not edit.
class LinkedList {
  constructor(value) {
    this.value = value;
    this.next = null;
  }
}

function mergeLinkedLists(headOne, headTwo) {
  // Write your code here.
}

// Do not edit the line below.
exports.LinkedList = LinkedList;
exports.mergeLinkedLists = mergeLinkedLists;

```

Our Tests

Custom Output Submit Code

```
}

addMany(values) {
  let current = this;
  while (current.next !== null) {
    current = current.next;
  }
  for (const value of values) {
    current.next = new LinkedList(value);
    current = current.next;
  }
  return this;
}

getNodesInArray() {
  const nodes = [];
```

Run or submit code when you're ready.