Our Solution(s) Run Code

Solution 1 Solution 2

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

package main

type LinkedList struct {
  value int
  next  *LinkedList
}

// 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
func MergeLinkedLists(headOne *LinkedList, headTwo *LinkedList) *Link
  recursiveMerge(headOne, headTwo, nil)
  if headOne.value < headTwo.value {
    return headOne
  }
  return headTwo
}

func recursiveMerge(p1, p2, p1Prev *LinkedList) {
  if p1 == nil {
    p1Prev.next = p2
    return
  }
  if p2 == nil {
    return
  }

  if p1.value < p2.value {
    recursiveMerge(p1.next, p2, p1)
    return
  }

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

// This is an input struct. Do not edit.
type LinkedList struct {
  Value int
  Next  *LinkedList
}

func MergeLinkedLists(headOne *LinkedList, headTwo *LinkedList) *Linke
  // Write your code here.
  return nil
}

```

Our Tests

Custom Output Submit Code

```
    if n == &LinkedList[Value: val]
    current = n
  for _, other := range others {
    current.Next = &LinkedList[Value: other]
    current = current.Next
  }
  return n
}

func (ll *LinkedList) Tokring() []int {
  vals := []int{}
  current := ll
  for current != nil {
    vals = append(vals, current.Value)
    current = current.Next
```

**Run or submit code when you're ready.**