

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4
5 using namespace std;
6
7 class LinkedList {
8 public:
9     int value;
10    LinkedList *next;
11
12    LinkedList(int value) {
13        this->value = value;
14        next = NULL;
15    }
16 };
17
18 void recursiveMerge(LinkedList *p1, LinkedList *p2, LinkedList *p1Prev
19
20 // O(n + m) time | O(n + m) space - where n is the number of nodes in
21 // Linked List and m is the number of nodes in the second Linked List
22 LinkedList *mergeLinkedLists(LinkedList *headOne, LinkedList *headTwo)
23     recursiveMerge(headOne, headTwo, NULL);
24     return headOne->value < headTwo->value ? headOne : headTwo;
25 }
26
27 void recursiveMerge(LinkedList *p1, LinkedList *p2, LinkedList *p1Prev
28     if (p1 == NULL) {
29         p1Prev->next = p2;
30         return;
31     }
32     if (p2 == NULL)
33         return;
```

Solution 1

Solution 2

Solution 3

```
1 #include <vector>
2
3 using namespace std;
4
5 // This is an input class. Do not edit.
6 class LinkedList {
7 public:
8     int value;
9     LinkedList *next;
10
11    LinkedList(int value) {
12        this->value = value;
13        next = NULL;
14    }
15 };
16
17 LinkedList *mergeLinkedLists(LinkedList *headOne, LinkedList *headTwo)
18     // Write your code here.
19     return NULL;
20 }
21
```

Our Tests

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4
5 using namespace std;
6
7 class LinkedList {
8 public:
9     int value;
10    LinkedList *next;
11
12    LinkedList(int value) {
13        this->value = value;
14        next = NULL;
15    }
16 };
17
18 void recursiveMerge(LinkedList *p1, LinkedList *p2, LinkedList *p1Prev
19
20 // O(n + m) time | O(n + m) space - where n is the number of nodes in
21 // Linked List and m is the number of nodes in the second Linked List
22 LinkedList *mergeLinkedLists(LinkedList *headOne, LinkedList *headTwo)
23     recursiveMerge(headOne, headTwo, NULL);
24     return headOne->value < headTwo->value ? headOne : headTwo;
25 }
```

```
1 #include <vector>
2
3 using namespace std;
4
5 // This is an input class. Do not edit.
6 class LinkedList {
7 public:
8     int value;
9     LinkedList *next;
10
11    LinkedList(int value) {
12        this->value = value;
13        next = NULL;
14    }
15 };
16
17 LinkedList *mergeLinkedLists(LinkedList *headOne, LinkedList *headTwo)
18     // Write your code here.
19     return NULL;
20 }
21
```

```
14     x_train = x_train_train
15
16     # Create a plot of the training data
17     plt.figure(figsize=(10, 5))
18     plt.scatter(x_train, y_train)
19     plt.title('Training Data')
20     plt.xlabel('x_train')
21     plt.ylabel('y_train')
22     plt.show()
23
24     # Create a plot of the test data
25     plt.figure(figsize=(10, 5))
26     plt.scatter(x_test, y_test)
27     plt.title('Test Data')
28     plt.xlabel('x_test')
29     plt.ylabel('y_test')
30     plt.show()
31
32     # Create a plot of the training data with the model's predictions
33     plt.figure(figsize=(10, 5))
34     plt.scatter(x_train, y_train)
35     plt.plot(x_train, y_train_hat)
36     plt.title('Training Data with Predictions')
37     plt.xlabel('x_train')
38     plt.ylabel('y_train')
39     plt.show()
40
41     # Create a plot of the test data with the model's predictions
42     plt.figure(figsize=(10, 5))
43     plt.scatter(x_test, y_test)
44     plt.plot(x_test, y_test_hat)
45     plt.title('Test Data with Predictions')
46     plt.xlabel('x_test')
47     plt.ylabel('y_test')
48     plt.show()
49
50     # Print the R-squared value for the training data
51     print('R-squared value for training data: %.2f' % r2_train)
```

Run or submit code when you're ready.