

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

Solution 2

```
1 # Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class LinkedList:
4     def __init__(self, value):
5         self.value = value
6         self.next = None
7
8
9 # O(n + m) time | O(n + m) space - where n is the number of nodes in the first
10 # Linked List and m is the number of nodes in the second Linked List
11 def mergeLinkedLists(headOne, headTwo):
12     recursiveMerge(headOne, headTwo, None)
13     return headOne if headOne.value < headTwo.value else headTwo
14
15
16 def recursiveMerge(p1, p2, p1Prev):
17     if p1 is None:
18         p1Prev.next = p2
19         return
20     if p2 is None:
21         return
22
23     if p1.value < p2.value:
24         recursiveMerge(p1.next, p2, p1)
25     else:
26         if p1Prev is not None:
27             p1Prev.next = p2
28         newP2 = p2.next
29         p2.next = p1
30         recursiveMerge(p1, newP2, p2)
31
```

Solution 1

Solution 2

Solution 3

```
1 # This is an input class. Do not edit.
2 class LinkedList:
3     def __init__(self, value):
4         self.value = value
5         self.next = None
6
7
8 def mergeLinkedLists(headOne, headTwo):
9     # Write your code here.
10     pass
11
```

Our Tests

Custom Output

Submit Code

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

```

18         current = current.next
19     # return the head of the linked list
20     return head
21
22 # Create a linked list
23 head = None
24
25 # Add nodes to the linked list
26 def addNode(value):
27     global head
28     # Create a new node
29     new_node = Node(value)
30     # If the list is empty, the new node is the head
31     if head is None:
32         head = new_node
33     # Otherwise, traverse to the end of the list
34     else:
35         current = head
36         while current.next is not None:
37             current = current.next
38         # Add the new node at the end
39         current.next = new_node
40
41 # Add nodes to the linked list
42 addNode(1)
43 addNode(2)
44 addNode(3)
45 addNode(4)
46 addNode(5)
47
48 # Print the linked list
49 def printList():
50     global head
51     current = head
52     while current is not None:
53         print(current.data, end=" ")
54         current = current.next
55     print()
56
57 # Print the linked list
58 printList()
59
60 # Run the program
61 if __name__ == '__main__':
62     addNode(1)
63     addNode(2)
64     addNode(3)
65     addNode(4)
66     addNode(5)
67     printList()

```

Run or submit code when you're ready.