Linked List 2

Q) Given 2 sorted linked list, merge them into a new linked list which is also sorted.

A = 5 → 8 → 10 → 12 → 14 **B** = 1 → 6 → 7 → 15

p1
5 → 8 → 10 → 12 → 14
head1

p2
1 → 6 → 7 → 15
head2

We will two pointers at A for ‘**`p1`**’ & ‘**`head1`**’ and for B, ‘**`p2`**’ & ‘**`head2`**’. We will check for maximum value between p1 & p2. Since p2 > p1, we will create a Node with value ‘1’ and this will be head3 and will put the pointer ‘cur’ to this value. After this step, since p2 value was taken, p2 pointer will move forward to 6.

Now, p1 = 5, p2 = 6. Since p1 < p2, we will assign the p1 to a new Node which will be and will move the ‘cur’ to the new Node.

Now, p1 = 8, p2 = 6. Since p1 > p2, we will assign the p2 to a new Node which will be and will move the ‘cur’ to the new Node.

Now, p1 = 8, p2 = 7. Since p1 < p2, we will assign the p2 to a new Node which will be and will move the ‘cur’ to the new Node.

Now, p1 = 8, p2 = 15. Since p1 < p2, we will assign the p1 to a new Node which will be and will move the ‘cur’ to the new Node.

Once p1 is exhausted, we will create another and assign p2 = 15 to it.

CODE:

We first need to create a linked list using class.

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

```python
def printList(head):
    cur = head
    while cur != None:
        print(cur.data, end = ' ')
        cur = cur.next

def merge(head1, head2):

    p1 = head1
    p2 = head2

    head3 = None
    cur = None

    while p1 != None and p2 != None:
        if p1.data < p2.data:
            if head3 is None:
                head3 = Node(p1.data)
                cur = head3
            else:
                cur.next = Node(p1.data)
                cur = cur.next
            p1 = p1.next
        else:
            if head3 is None:
                head3 = Node(p2.data)
                cur = head3
            else:
                cur.next = Node(p2.data)
                cur = cur.next
            p2 = p2.next

    while p1 != None:
        cur.next = Node(p1.data)
        cur = cur.next
        p1 = p1.next

    while p2 != None:
        cur.next = Node(p2.data)
        cur = cur.next
        p2 = p2.next

    return head3
```

```python
if __name__ == "__main__":

    head1 = Node(5)
    head1.next = Node(15)
    head1.next.next = Node(25)
    head1.next.next.next = Node(125)

    print("head1 list")
    printList(head1)

    head2 = Node(1)
    head2.next = Node(3)
    head2.next.next = Node(6)
    head2.next.next.next = Node(9)
    head2.next.next.next.next = Node(19)

    print()
    print("head2 list")
    printList(head2)
    print()

    head3 = merge(head1, head2)
    print()
    print("merge list is:")
    printList(head3)
```

Q) Given a linked list, reverse it

given = 1 → 2 → 3 → 4 → 5

res = 5 → 4 → 3 → 2 → 1

1 → 2 → 3 → 4 → 5
head

5 → 4 → 3 → 2 → 1
head

3 Pointer Technique:

We have to reverse the link. We will be using a ‘prev’, ‘cur’ and ‘next’ pointers for this method.

Initially, prev will be None, with cur at the head of the linked list and next will be at 2nd position. So,

```
next = cur.next
cur.next  = prev
prev = cur
cur = next
```

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

def reverseLinkedList(head):
    prev = None
    cur = head
    while cur != None:
        next = cur.next
        cur.next = prev
        prev = cur
        cur = next
    return prev

if __name__ == "__main__":

    head1 = Node(5)
    head1.next = Node(15)
    head1.next.next = Node(25)
    head1.next.next.next = Node(125)


    reverse_list = reverseLinkedList(head1)
    print("reverse list is:")
    printList(reverse_list)
```