Assignment 9 (Linked List)

> **Name: Mohammed Varaliya**

> **Roll No: 54**

Questions

1. **Code of Recursive function for counting number of nodes present in a linked list.**
2. **Code of Recursive function for calculating total of all node values in a linked list.**

---

1. **Code of Recursive function for counting number of nodes present in a linked list.**
   a. **Code:**

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

class SinglyLinkedList:
```

```
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

    def prepend(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        cur_node = self.head
        self.head = new_node
        new_node.next = cur_node

    def print_list(self):
        cur_node = self.head
        while cur_node:
            print(cur_node.data, end=" -> ")
            cur_node = cur_node.next
        print("None")

    def len_recursive(self, node):
        if node is None:
            return 0
        return 1 + self.len_recursive(node.next)
```

```
if __name__ == "__main__":

    llist = SinglyLinkedList()
    llist.append(1)
    llist.append(2)
    llist.append(3)
    llist.append(4)

    llist.print_list()

    print(llist.len_recursive(llist.head))
```

```
# Main logic of Recursive function for counting numbe
r of nodes present in a linked list in above code is.

def len_recursive(self, node):
    if node is None:
        return 0
    return 1 + self.len_recursive(node.next)
```

1. **Explanation:**
    a. **Purpose**: This function counts the total number of nodes in a singly linked list using recursion.
    b. **Logic**: The function `len_recursive` traverses the list starting from the head node. If the current node is `None` (end of the list), it returns `0`. Otherwise, it returns `1 + len_recursive(node.next)` to count the node and move to the next node.
    c. **Output**: Given a list of nodes `1 -> 2 -> 3 -> 4`, the function returns `4`, indicating the total number of nodes.
    d. **Example**: For the list `1 -> 2 -> 3 -> 4`, it prints `1 -> 2 -> 3 -> 4 -> None` and then outputs `4`.

b. **Output:**

```
1 -> 2 -> 3 -> 4 -> None
```

```
4
```

---

2. **Code of Recursive function for calculating total of all node values in a linked list.**
    a. **Code:**

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

class SinglyLinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

    def prepend(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        cur_node = self.head
        self.head = new_node
```

```
            new_node.next = cur_node

        def print_list(self):
            cur_node = self.head
            while cur_node:
                print(cur_node.data, end=" -> ")
                cur_node = cur_node.next
            print("None")

        def sum_recursive(self, node):
            if node is None:
                return 0
            return node.data + self.sum_recursive(node.next


    if __name__ == "__main__":

        llist = SinglyLinkedList()
        llist.append(1)
        llist.append(2)
        llist.append(3)
        llist.append(4)

        llist.print_list()

        print(llist.sum_recursive(llist.head))
```

1. **Explanation:**
    a. **Purpose**: This function calculates the sum of all node values in a singly linked list using recursion.
    b. **Logic**: The function `sum_recursive` starts from the head node and adds the value of each node to the result of the recursive call on the next node. If it encounters `None`, it returns `0`.
    c. **Output**: For a list `1 -> 2 -> 3 -> 4`, it returns `10`, the sum of all node values.
    d. **Example**: For the list `1 -> 2 -> 3 -> 4`, it prints `1 -> 2 -> 3 -> 4 -> None` and then outputs `10`.

```
# Main logic of Recursive function for calculating to
tal of all node values in a link list is.

def sum_recursive(self, node):
    if node is None:
        return 0
    return node.data + self.sum_recursive(node.next)
```

b. **Output:**

```
1 -> 2 -> 3 -> 4 -> None
10
```
