

To remove duplicates from a sorted linked list and return the list with each element appearing only once, you can implement a function that iterates through the list and modifies the pointers accordingly. Below is the implementation in Python.

Python Implementation:

```
class ListNode:

    def __init__(self, val=0, next=None):

        self.val = val

        self.next = next

def deleteDuplicates(head: ListNode) -> ListNode:

    current = head

    while current and current.next:

        if current.val == current.next.val:

            # Skip the duplicate node

            current.next = current.next.next

        else:

            # Move to the next node

            current = current.next

    return head
}
```

Explanation:

1. Class Definition: The `ListNode` class represents a node in the linked list.
2. Function Definition: The `deleteDuplicates` function takes the head of the linked list as an argument.

3. Iteration:

- A while loop iterates through the linked list.
- If the current node's value is the same as the next node's value, it skips the next node by updating the next pointer of the current node.
- If they are different, it simply moves to the next node.

4. Return: Finally, the modified head of the list is returned.

Example 1

```
head1 = ListNode(1, ListNode(1, ListNode(2)))
```

```
result1 = deleteDuplicates(head1)
```

```
printList(result1) # Output: 1 1 2
```

Example 2

```
head2 = ListNode(1, ListNode(1, ListNode(2, ListNode(3, ListNode(3)))))
```

```
result2 = deleteDuplicates(head2)
```

```
printList(result2) # Output: 1 2 3
```

This implementation efficiently removes duplicates from a sorted linked list while maintaining its order.