Copilot

Actually, in the `printList` function, there's no need to use a double pointer. The reason is that we're not modifying the `head` pointer itself, but only using it to traverse the list. When we do `node = node->next;` in the `printList` function, we're only changing the local `node` pointer, not the `head` pointer in the list. The `head` pointer in the list remains unchanged, so it will still point to the first node even after `printList` is called.

Here's the `printList` function for reference:

```
void printList(Node* node) {
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
}
```

In this function, `node` is a copy of the `head` pointer. Changing `node` does not change `head`. So, after this function is called, `head` will still point to the first node of the list, not the last node.

Double pointers are useful when we want to change the pointer itself (like in `insert` and `freeLinkedList` functions where we want to change `head`). But for simply accessing or traversing the list (like in `printList`), a single pointer is sufficient. Hope this clarifies your doubt! 😊