Copilot

Absolutely, it's a good practice to free the memory that was allocated dynamically when it's no longer needed. Here's how you can add a `freeLinkedList` function to the code:

```
#include <stdio.h>
#include <stdlib.h>

// Define the Node data structure
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Define the LinkedList data structure
typedef struct LinkedList {
    Node* head;
} LinkedList;

// Function to insert a node at the end of the linked list
void insert(LinkedList* list, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;

    if (list->head == NULL) {
        list->head = newNode;
    } else {
        Node* last = list->head;
        while (last->next != NULL) {
            last = last->next;
        }
        last->next = newNode;
    }
}

// Function to print the linked list
void printList(LinkedList* list) {
    Node* node = list->head;
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
}

// Function to free the linked list
void freeLinkedList(LinkedList* list) {
    Node* current = list->head;
    Node* next;

    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }

    list->head = NULL;
}

int main() {
    LinkedList list;
    list.head = NULL;

    insert(&list, 1);
    insert(&list, 2);
    insert(&list, 3);
    insert(&list, 4);
    insert(&list, 5);

    printList(&list);
```

```
    freeLinkedList(&list);

    return 0;
}
```

In this version of the code, the `freeLinkedList` function traverses the list, freeing each node as it goes. After all nodes have been freed, it sets the `head` pointer of the list to `NULL`. This function is called in the `main` function after the list has been printed. This ensures that the memory allocated for the list is freed before the program exits. Remember, it's important to avoid memory leaks in your programs, especially when working with dynamically allocated memory. This is a good practice in C programming.