Doubly Linked List Assignment

Write a Python program that uses (imports) the DLL class we constructed in class to implement a solution to a problem similar to the textbook's exercise P-7.44 (although I've changed/clarified a few details). Specifically, write a class named TextEdit that provides the following methods for editing a line of text:

- `insert(c)`: Inserts character c at the current location of the cursor. The cursor shifts one position to the right in the text string, so that the inserted character is immediately to the left of the cursor.
- `delete()`: Deletes the character at the current location of the cursor. The cursor stays at the same position in the text string. Does nothing if the cursor is past the last character in the text string.
- `left()`: Move the cursor to the left one position. Do not move the cursor if it is on the first character of the text string or if the string is empty.
- `right()`: Move the cursor to the right one position. Do not move the cursor if it is already past the last character in the text string.
- `print()`: Prints the current text string, then prints a line underneath that consists of spaces and a single caret (^) at the location of the cursor in the text string.

Initially, the text string has no characters and the cursor is conceptually located at the position one character after this string (at the first character of the line).

For instance, the following code

```
t = TextEdit()
t.insert('a')
t.insert('b')
t.print()
t.left()
t.print()
t.insert('c')
t.insert('d')
t.print()
t.delete()
t.delete()
t.print()
for i in range(4):
  t.left()
t.delete()
t.print()
```

should produce the following output:

```
ab
  ^
ab
 ^
acdb
   ^
acd
   ^
cd
^
```

Details/Hints

- Your program must implement the TextEdit class by creating a DLL object and, to implement the `insert()` and `delete()` methods, calling the `insert_between()` and `delete_node()` methods on this object. That is, your program should in this way be similar to the solutions to the Singly and Doubly Linked List POGIL applications posted at Blackboard, which use calls on SLL and DLL objects to provide the functionality of the Stack, Queue, and Deque ADTs.
- You are allowed to use my implementation of the DLL class, posted at Blackboard, or your own implementation. Note that my implementation initializes the `header` and `trailer` nodes to point to each other through `header.next` and `trailer.prev`, which I found helpful when writing my solution.
- In my implementation, in addition to a DLL object, my class defines a `_cursor` attribute that initially is equal to the `trailer` of the DLL object. I "move" the cursor left or right by setting this attribute to its `prev` or `next` value as appropriate. This approach is not required, but again, I found it helpful.
- If you're unclear about how any of the methods are supposed to operate, please ask.