Linked List Algorithm Practice

Announcements

Re: Assignments:

- **EX07: Linked List Utility Functions** due Tues. Nov 25th

Re: Quiz 04:

- Quiz practice is on the site!
- If you want help:
  - Office Hours are open 11am-5pm every day before the quiz
  - Tutoring is available 5-7pm on Monday, Wednesday, and Thursday
  - Hybrid Review Session at 5:30pm on Thursday in Fred Brooks 141

`insert_after` Algorithm Demo

1. When you are asked,
   "Can you add a Node with a value of 211 after the Node with value 210?"

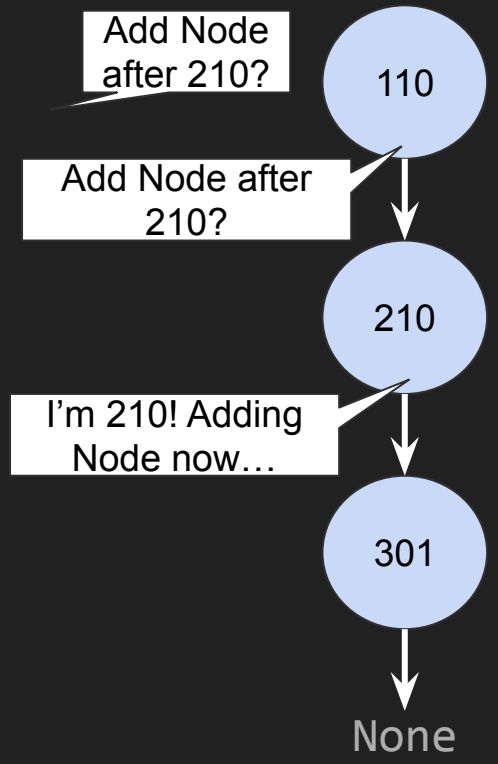
If your value ***is not 210***:

2. Ask the ***next*** Node,
   "Can you add a Node with a value of 211 after the Node with value 210?"
   Wait patiently for an answer!

3. Once the answer is returned back to you, turn to the person who asked you and give them this answer.

If your value **is 210**:

2. Invite a new friend to the list! You will now point to them, and they will point to the person you were previously pointing to. New Node, you'll say "I was added!!"



`insert_after` Algorithm Demo

1. When you are asked,
   "Can you add a Node with a value of 211 after the Node with value 210?"

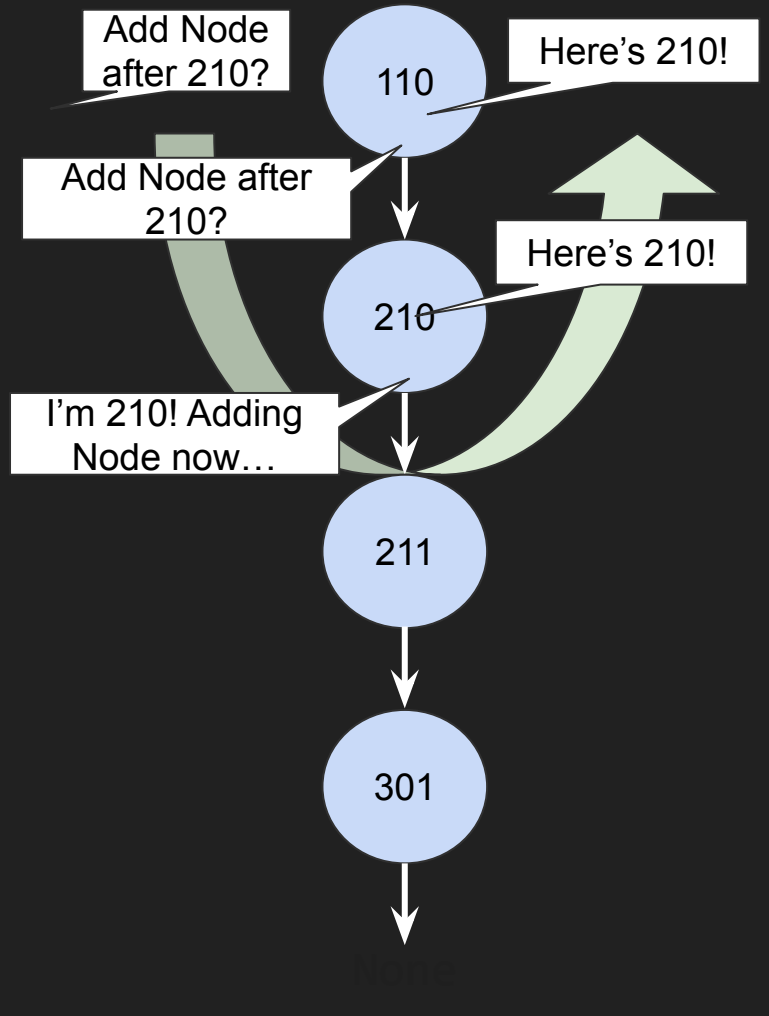
If your value ***is not 210***:

2. Ask the ***next*** Node,
   "Can you add a Node with a value of 211 after the Node with value 210?"
   Wait patiently for an answer!

3. Once the answer is returned back to you, turn to the person who asked you and give them this answer.

If your value **is 210**:

2. Invite a new friend to the list! You will now point to them, and they will point to the person you were previously pointing to. New Node, you'll say "I was added!!"



Let’s write pseudocode for the `insert_after` function

Let’s write the `insert_after` function in VS Code! 

```python
from __future__ import annotations

class Node:
    """Node in a singly-linked list recursive structure."""

    value: int
    next: Node | None

    def __init__(self, value: int, next: Node | None):
        self.value = value
        self.next = next

    def __str__(self) -> str:
        if self.next is None:
            return f"{self.value} -> None"
        else:
            return f"{self.value} -> {self.next}"

def insert_after(head: Node | None, val_to_add: int, search_val: int) -> Node:
    if head is None:
        raise ValueError("Value does not exist in linked list.")
    if head.value == search_val:
        new_node: Node = Node(val_to_add, head.next)
        head.next = new_node
        return new_node
    else:
        return insert_after(head.next, val_to_add, search_val)

courses: Node = Node(210, Node(301, None))
print(courses)
insert_after(courses, 211, 210)
print(courses)
```