Mod 4 Homework - Extended LinkedLists

Two important notes regarding this assignment:

1) This homework also serves as practice for the first exam - we have asked all of these questions on the programming portion of exam 1 (with some modifications, like using a Doubly Linked List instead of a Linked List) in previous semesters.

2) This assignment is 100% autograded, so you get immediate feedback on your performance while preparing for the exam. As always, feel free to discuss the assignment conceptually with your classmates, but note that sharing code is academic misconduct - keep the discussions conceptual, and never share code.

Starter Code

We may provide some basic starter code on the exam and ask you to extend it. For this assignment, you can use the `Node` and `LinkedList` classes from the Mod 4 lab as your starter code (see the diagrams below). We will be extending these classes in this assignment - go ahead and create them if you don't have them already.

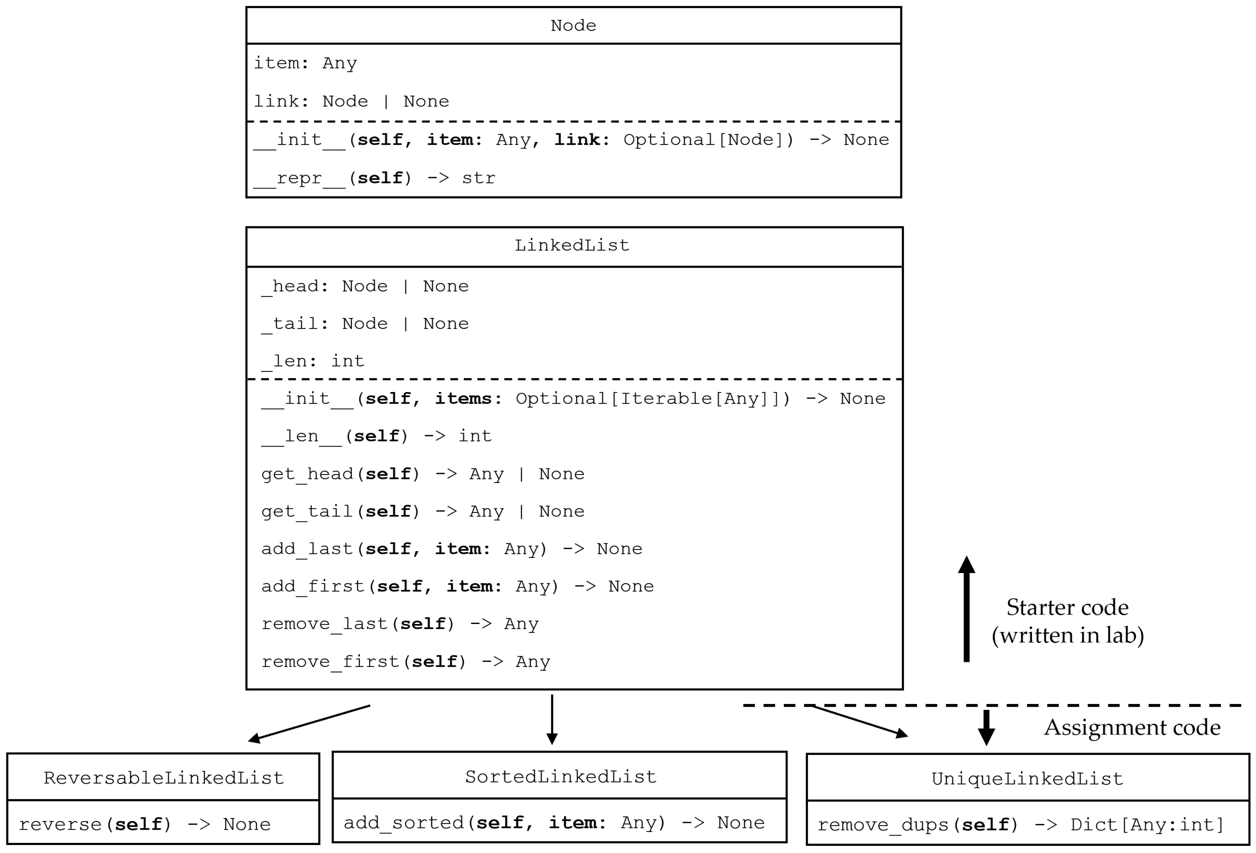

Node
item: Any
link: Node | None
__init__(self, item: Any, link: Optional[Node]) -> None
__repr__(self) -> str
LinkedList
_head: Node | None
_tail: Node | None
_len: int
__init__(self, items: Optional[Iterable[Any]]) -> None
__len__(self) -> int
get_head(self) -> Any | None
get_tail(self) -> Any | None
add_last(self, item: Any) -> None
add_first(self, item: Any) -> None
remove_last(self) -> Any
remove_first(self) -> Any
Starter code (written in lab)
Assignment code
ReversableLinkedList
reverse(self) -> None
SortedLinkedList
add_sorted(self, item: Any) -> None
UniqueLinkedList
remove_dups(self) -> Dict[Any:int]

Figure 1: Starter code (written in lab) and the classes you'll be extending it to in this assignment.

Question 1: Reversable LinkedList

Create a ReversableLinkedList class that **extends** your LinkedList class. In object-oriented programming, “extends” means you inherit from that class and add some functionality via starter attributes or new methods. The only method you need add to to the public interface is reverse(). This method should reverse the direction of the Linked List by redirecting all link attributes. Note that you **should not** create any new objects (no new nodes or lists) inside this method - just relink the nodes you already have, making sure to update _head and _tail as appropriate:

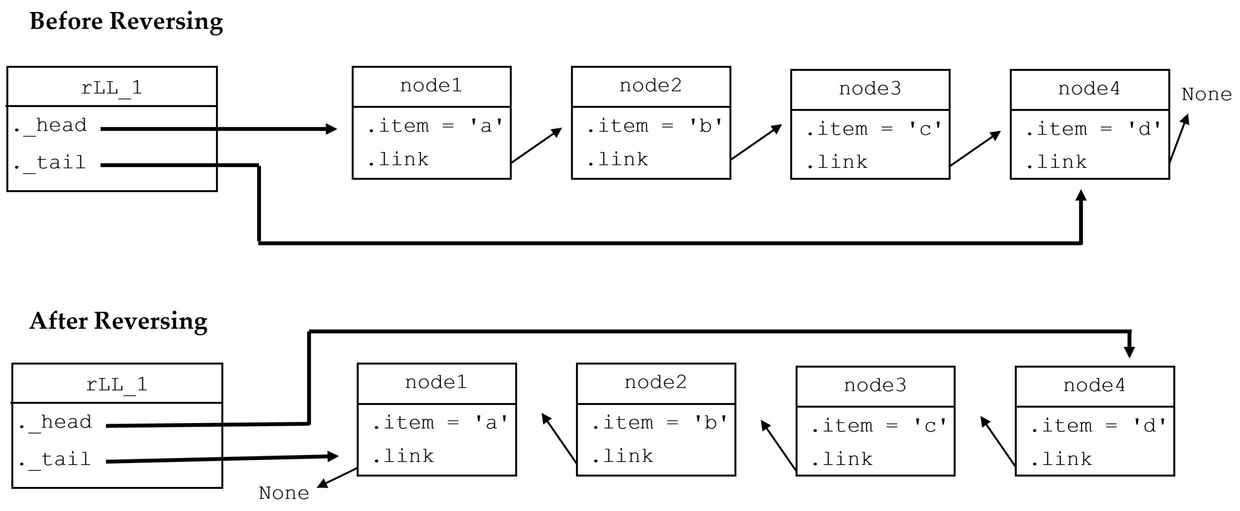


Figure 2: The same ReversableLinkedList object rLL_1 before and after reversing. All that changes are node pointers: _head, _tail, and all .link attributes. No new node objects are created.

reverse() should be O(n).

Examples

```
>>> rLL1 = ReversableLinkedList('abcd')
>>> isinstance(rLL1, ReversableLinkedList), isinstance(rLL1, LinkedList)
(True, True)
>>> rLL1.get_head(), rLL1.get_tail() # head->a->b->c->d<-tail
('a', 'd')
>>> rLL1.reverse()
>>> rLL1.get_head(), rLL1.get_tail() # tail->a<-b<-c<-d<-head
('d', 'a')
>>> while rLL1: # rLL1 evaluates as True until empty becuase it provides __len__
        print(rLL1.remove_first())
...
d
c
b
a
```

Question 2: SortedLinkedList

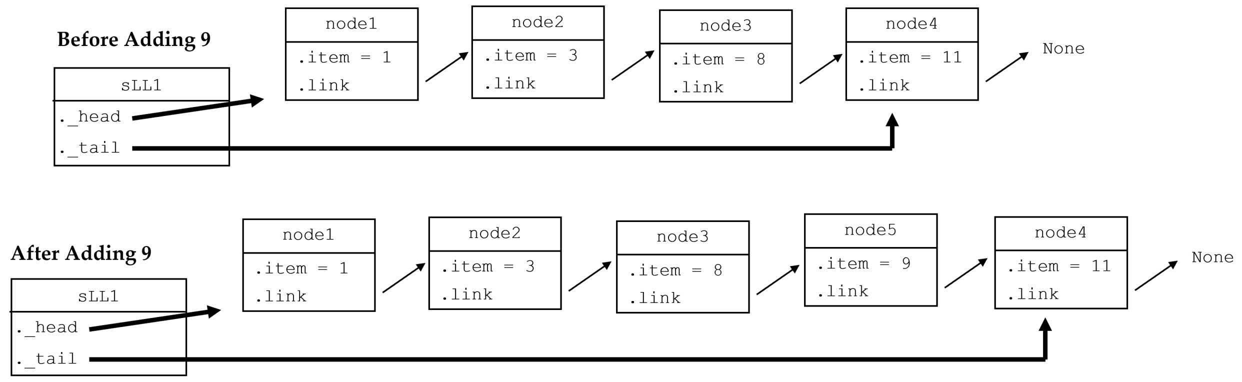
Create a class `SortedLinkedList` that extends `LinkedList` with the additional method `add_sorted(item)`. When `add_sorted()` is called, iterate through the Linked List until you find the first item greater than or equal to the item being added, and insert the new node just before it.

We should restrict the public interface for adding items to just `add_sorted()` - raise a `NotImplementedError` if someone tries to call `add_first` or `add_last` by **overloading** those methods:

```
class SortedLinkedList(LinkedList):
    ...
    def add_first(self, item):
        raise NotImplemntedError(f"Use add_sorted({item}) instead")
```

Note that you should only create one node every time this method is called (rather than just copying the whole list to new nodes.)

`add_sorted()` should be `O(n)`.



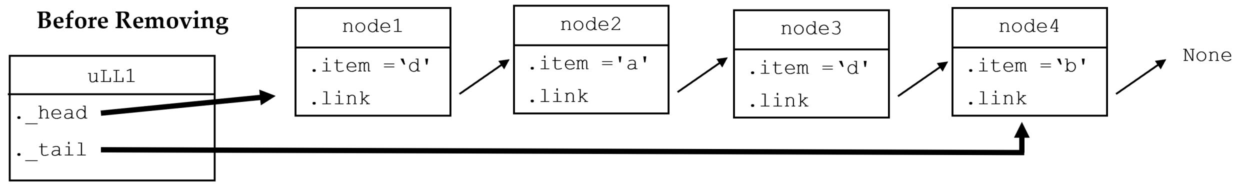
Examples

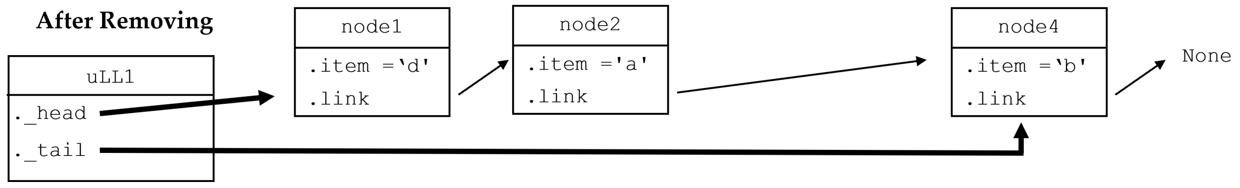
```
>>> sLL1 = SortedLinkedList()
>>> isinstance(sLL1, SortedLinkedList), isinstance(sLL1, LinkedList)
(True, True)

>>> sLL1.add_first(8) # don't want this in the public interface anymore
...
NotImplementedError
>>> sLL1.add_sorted(8); sLL1.add_sorted(3); sLL1.add_sorted(11); sLL1.add_sorted(1)
>>> sLL1.add_sorted(9)
>>> while sLL1: # sLL1 evaluates as true until empty becuase it provides __len__
        print(sLL1.remove_first())
...
1
3
8
9
11
```

Question 3: UniqueLinkedList

Write a class `UniqueLinkedList` that extends `LinkedList` and adds an additional method - `remove_dups()`. This method should remove any nodes with duplicate items from your `LinkedList`, only keeping the earliest node with that item. Return a dictionary of `item:count` representing how many copies of each item were removed, including a count of 0 for any items that only occured once. Note that you **should not** create any new new nodes or linked lists inside this method - just relink the nodes you already have.

When the function ends, your linked list should not have any duplicates:





`remove_dups()` should be `O(n)`

Examples

```
>>> uLL1 = UniqueLinkedList(['d', 'a', 'd', 'b'])
>>> is_instance(uLL1, UniqueLinkedList)
True
>>> is_instance(ull1, LinkedList)
True
>>> len(uLL1)
4
>>> uLL1.get_head()
'd'
>>> uLL1.get_tail()
'b'
>>> print(ull1.remove_dups())
{'d':1, 'a':0, 'b':0}
>>> while uLL1: # uLL1 evaluates as True until empty because it provides __len__
       print(uLL1.remove_first())
d
a
b
```

Imports

No imports are allowed on this assignment, with the following exceptions:

- `LinkedList` and `Node` classes from `linkedlist.py`
- `typing` (not required, but you're welcome to use it if you'd like)
- For testing only (do not use these for functionality in any other classes/algorithms)
  - unittest
  - random

Submission Requirements

Students must submit **individually** by the due date (Tuesday, February 20th at 11:59 pm EST) to receive credit. This deadline is after the first exam, but it will help you immensely to have solved these problems before the exam.

At a minimum, you must submit the following files:

- From Mod 4 Lab:
  - `linkedlist.py`
    - `class Node`
    - `class LinkedList`
- Written for this assignment:
  - `extendedlinkedlists.py`
    - `class ReversableLinkedList(LinkedList)`
    - `class SortedLinkedList(LinkedList)`
    - `class UniqueLinkedList(LinkedList)`

Additionally, you must include any other files necessary for your code to run, such as modules containing data structures you wrote yourself.