NAME _____ **ICA-10**

NetID_____ @email.arizona.edu

Work with your neighbor.

Problem 1. Implement two classes with the following methods

```
class Node:
    def __init__(self, value):

    def value(self):
        """getter for the _value attribute"""

    def next(self):
        """getter for the _next attribute"""

    def __str__(self):
```

```
class LinkedList:
    def __init__(self):
        """initialize two attributes _head and
_size"""

    def add(self, node):
        """add a single node to the head"""

    def __len__(self):
```

Run the program and create a linked list as shown in the following IDLE transcript:

```
>>> a_ll = LinkedList()
>>> len(a_ll)
0
>>> node1 = Node("world")
>>> print(node1)
world
>>> node2 = Node("hello")
>>> a_ll.add(node1)
>>> len(a_ll)
1
>>> a_ll.add(node2)
>>> len(a_ll)
2
```

Problem 2. Add `print_nodes` method to `LinkedList` class, which prints out all nodes' value in the linked list. Following the above IDLE transcript, you need to achieve this:

```
>>> a_ll.print_nodes()
hello
world
```

Problem 3. Add `append(self, node)` method to `LinkedList` class, that appends a node to the tail of the list. To implement it effeciently, maintaining `_tail` attribute in `LikedList`. Following the above IDLE transcript, you need to achieve this:

```
>>> a_ll.append(Node("!!!"))
>>> a_ll.print_nodes()
hello
world
!!!
>>> b_ll = LinkedList()
>>> b_ll.append(Node("happy"))
>>> len(b_ll)
1
>>> b_ll.print_nodes()
happy
>>> b_ll.append(Node("life"))
>>> len(b_ll)
2
>>> b_ll.print_nodes()
happy
life
```

Problem 4. Add `remove_first(self)` method to `LinkedList` class, that removes the first node in the list, and returns the node. The removed `node._next` should be `None`. If an LinkedList object is empty, return None. Following the above IDLE transcript, you need to achieve this:

```
>>> n1 = b_ll.remove_first()
>>> print(n1)
happy
>>> n1.next() == None
True
>>> b_ll.print_nodes()
life
>>> b_ll.remove_first()
<linkedlist.Node object at 0x1026ed438>
>>> len(b_ll)
0
>>> b_ll.remove_first()
>>>
```

Problem 5. Add `concat(self, another)` method to `LinkedList` class, that concats another list to the current list. Following the above IDLE transcript, you need to achieve this:

```
>>> c_ll = LinkedList()
>>> c_ll.append(Node("This"))
>>> c_ll.append(Node("is"))
>>> c_ll.append(Node("cool"))
```

```
>>> a_ll.concat(c_ll)
>>> a_ll.print_nodes()
hello
world
!!!
This
is
cool
>>> a_ll.append(Node("!!!!!"))
>>> a_ll.print_nodes()
hello
world
!!!
This
is
cool
!!!!!
```

Problem 6 (optional). Add `reverse` method to `LinkedList` class, that reverse the linkedlist.