

CSC148: Linked Lists

Daniel Zingaro
University of Toronto

Here is a linked list node for the following questions. The empty linked list will be represented as None.

```
class Node:

    def __init__(self, value):
        self.value = value
        self.next = None
```

Question 1

Let's practice with this linked list node. Show the linked list that is created by the following code.

```
a = Node(4)
a.next = Node(5)
b = Node(6)
c = a.next
c.next = b
```

Question 2

Explain in plain English the purpose of each of the following functions.

```
def mystery1(lnk: Node): # purpose???
    while lnk and lnk.next:
        lnk.next = lnk.next.next
        lnk = lnk.next

def mystery2(lnk: Node) -> Node: # purpose???
    if not lnk:
        return
    lnk = lnk.next
    start = lnk
    while lnk and lnk.next:
        lnk.next = lnk.next.next
        lnk = lnk.next
    return start
```

Question 3

To **reverse** a linked list means to reverse the order of its elements; for example, 1->2->3 would become 3->2->1.

The following is an attempt to write a function to reverse a linked list. Provide a counterexample showing that the function is **not** correct, or give informal “proof” that the function **is** correct.

```
def reverse(lnk: Node) -> Node:
  reversed = None
  nonreversed = lnk
  while nonreversed:
    rest = nonreversed.next
    nonreversed.next = reversed
    reversed = nonreversed
    nonreversed = rest
  return reversed
```

Question 4

Write the following function that takes two linked lists `lst1` and `lst2` and merges the elements of `lst2` into `lst1`. i.e. if `lst1` is 1 -> 2 -> 3 and `lst2` is 4 -> 5 -> 6, then

`merge(lst1, lst2)` makes

`lst1` be 1 -> 4 -> 2 -> 5 -> 3 -> 6

`lst2` is unchanged. Assume that `lst2` has no more elements than `lst1`.

```
def merge(lst1: Node, lst2: Node) -> None:
```