- This lab will cover Linked Lists.
- It is assumed that you have reviewed chapter 6 & 7 of the textbook. You may want to refer to the text and your lecture notes during lab as you solve the problems.
- When approaching the problems, think before you code. Doing so is good practice and can help you lay out possible solutions.
- Think of any possible test cases that can potentially cause your solution to fail!
- You must stay for the duration of the lab. If you finish early, you may help other students. If you don't finish by the end of the lab, we recommend you complete it on your own time.
- Your TAs are available to answer questions in lab, during office hours, and on Piazza.

Vitamins (maximum 20 minutes)

---

1. Draw the memory image of the linked list object as the following code executes:

```
from DoublyLinkedList import DoublyLinkedList


dll = DoublyLinkedList()
dll.add_first(1)
1
dll.add_last(3)
1 <-> 3
dll.add_last(5)
1 <-> 3 <-> 5
dll.add_after(dll.header.next, 2)
1 <-> 2 <-> 3 <-> 5
dll.add_before(dll.trailer.prev, 4)
1 <-> 2 <-> 3 <-> 4 <-> 5
dll.delete_node(dll.trailer.prev)
1 <-> 2 <-> 3 <-> 4
dll.add_first(0)
0 <-> 1 <-> 2 <-> 3 <-> 4

print(dll)
```

What is the output of the code?

```
0 <-> 1 <-> 2 <-> 3 <-> 4
```

2. During lecture you learned about the different methods of a doubly linked list.

   Provide the following worst-case runtime for those methods:

   a. `def __len__(self):`
   Θ(1)

   b. `def is_empty(self):`
   Θ(1)

   c. `def add_after(self, node, data):`
   Θ(1)

   d. `def add_first(self, data):`
   Θ(1)

   e. `def add_last(self, data):`
   Θ(1)

   f. `def add_before(self, node, data):`
   Θ(1)

   g. `def delete_node(self, node):`
   Θ(1)

   h. `def delete_first(self):`
   Θ(1)

   i. `def delete_last(self):`
   Θ(1)

3. Trace the following function. What is the output of the following code? Give mystery an appropriate name.

```
#dll = Doubly Linked List
def get_and_remove_second_last(dll):

    if len(dll) >= 2:
        node = dll.trailer.prev.prev #gets second last
        node.prev.next = node.next   #removes
        node.next.prev = node.prev

        node.next = None
        node.prev = None
        return node

    else:
        raise Exception("dll must have length of 2 or
        greater")

print(mystery(dll))
```

Input:
1 <-> 2 <-> 3 <-> 4
Output:
1 <-> 2 <-> 4

NOTE when removing a node, we must change 4 pointers in a DLL