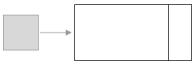
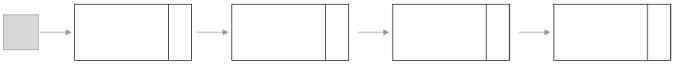
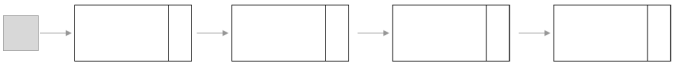
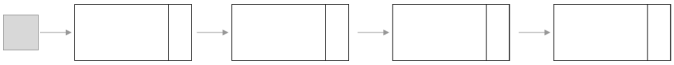
CSC148 Worksheet 12 Solution

Hyungmo Gu

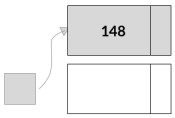
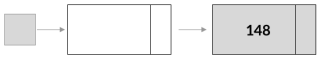
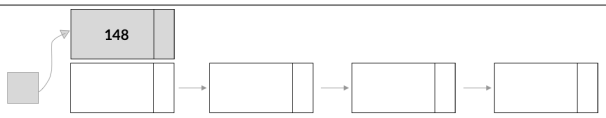
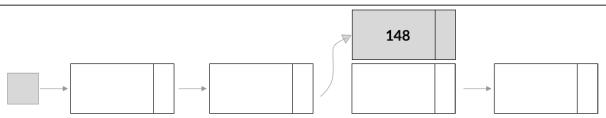
April 23, 2020

Question 1

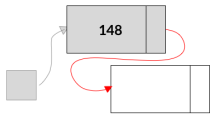
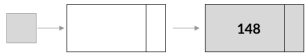
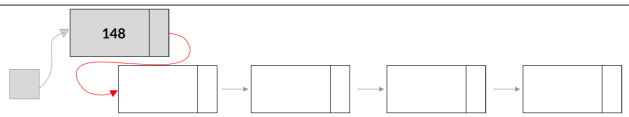
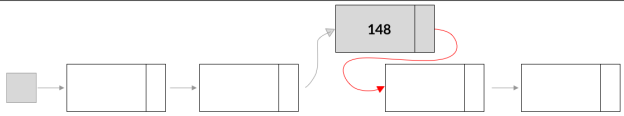
a.

Input description	Linked list diagram
`len(self) == 0, index == 0`	
`len(self) == 1, index == 0`	
`len(self) == 1, index == 1`	
`len(self) == 4, index == 0`	
`len(self) == 4, index == 2`	
`len(self) == 4, index == 4`	

b.

Input description	Linked list diagram
`len(self) == 0, index == 0`	
`len(self) == 1, index == 0`	
`len(self) == 1, index == 1`	
`len(self) == 4, index == 0`	
`len(self) == 4, index == 2`	
`len(self) == 4, index == 4`	

Correct Solution:

Input description	Linked list diagram
`len(self) == 0, index == 0`	
`len(self) == 1, index == 0`	
`len(self) == 1, index == 1`	
`len(self) == 4, index == 0`	
`len(self) == 4, index == 2`	
`len(self) == 4, index == 4`	

Question 2

a. To reassign *self._first* to something new, *len(self)* can be value, but *index* has to be at 0.

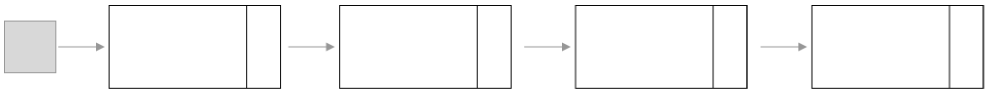
b. To make *insert* method to behave the same as *LinkedList.append*, *len(self)* can be any value, but $index = len(self) - 1$.

Correct Solution:

To make *insert* method to behave the same as *LinkedList.append*, *len(self)* can be any value, but $index = len(self)$.

Notes:

- Learned that the first node in linked list represents $index = 0$.



- Learned that *insert* operation for linked list is the same as the *insert* operation for lists.
  - If node doesn't exist at this index, then hook last node to the inserting node.
  - If node does exist, then hook current node to one end of the inserting node, and the other end to the next node

Question 3

```
    ...
    def insert(self, index: int, item: Any) -> None:
        """Insert a the given item at the given index in this list.
        Raise IndexError if index > len(self) or index < 0.
        Note that adding to the end of the list is okay.

        >>> lst = LinkedList([1, 2, 3])
        >>> lst.insert(0,0)
        >>> str(lst)
        '[0 -> 1 -> 2 -> 3]'
        >>> lst.insert(1,10)
        >>> str(lst)
        '[0 -> 10 -> 1 -> 2 -> 3]'
        >>> lst.insert(5,10)
        >>> str(lst)
        '[0 -> 10 -> 1 -> 2 -> 3 -> 10]'
        """

        curr = self._first
        current_index = 0

        if index == 0:
            current_node = curr
            self._first = _Node(item)
            self._first.next = current_node
            return

        while curr is not None:
            if current_index != index - 1:
                curr = curr.next
                current_index += 1
                continue

            # 1. if index == len(self), then append to list
            # 2. if index < len(self), then insert value at index
            current_node = curr.next
            inserting_node = _Node(item)
```

```
            curr.next = inserting_node
            inserting_node.next = current_node
            return

        # 3. if index > len(self), raise IndexError
        raise IndexError
    ...
```

Listing 1: worksheet_12_q3_solution.py