Assignment One – Palindrome Check

Timothy Polizzi
Timothy.Polizzi1@Marist.edu

February 12, 2019

1 MyStack

```
# A stack for project1 of CMPT435.

__author__ = "Tim Polizzi"
__email__ = "Timothy.Polizzi1@marist.edu"

from Assignments.AssignmentOne.MyLinkedList import MyLinkedList


class MyStack(MyLinkedList):
    """A stack generated without the assistance of external
    libraries for CMPT435.

    A stack generated separate from any pre-existing libraries, for
     the purpose of learning how to
    use stacks for Allan Labouseur's algorithms class.
    """
    def __init__(self):
        """Initializes MyStack"""
        super().__init__()
        self.inner_list = MyLinkedList()

    def push(self, to_push: str):
        """Adds an item to MyStack.

        Appends an item to the top of MyStack.

        Args:
            to_push: The item that is to be appended to MyStack.
        """
```

```
28          self.inner_list.new_node = self.inner_list.TextNode(to_push
    , self.inner_list.head)
29          self.inner_list.head = self.inner_list.new_node
30
31      def pop(self) -> str:
32          """Removes an item from MyStack.
33
34          Removes the topmost item from MyStack and returns its value
    .
35
36          Returns:
37              The value of the item that was removed from MyStack.
38          """
39          if self.inner_list.is_empty():
40              raise Exception('List is empty')
41          else:
42              old_head_val = self.inner_list.head.val
43              self.inner_list.head = self.inner_list.head.next
44              return old_head_val
45
46      def peek(self) -> str:
47          """Returns the value of the item on top of MyStack."""
48          peek_val = self.pop
49          self.push(str(peek_val))
50          return str(peek_val)
```

The implementation of MyStack was done through the usage of a LinkedList that I generated. MyStack used the LinkedList in order to maintain positions of the items in the stack, but MyStack implemented all pop and push methods fully even though the LinkedList already had implementations that could be used.

1.1 Push (20-29)

The push method, used to add a new node to the top of MyStack, specifically takes in a string which it then uses to generate a new TextNode which is given a pointer to the old head of MyStack.

1.2 Pop (31-44)

The pop method of MyStack, which is used to remove the top-most value of MyStack and returns its value, initially checks if the stack is empty to make sure it is not attempting to remove from an empty stack. Then it locally saves the the value of the old head and then sets the head to the next item in MyStack.

1.3 Peek (46-50)

The peek method is intended to look at the top-most item in MyStack and return it. It does so by running pop, saving the value and then pushing that back onto MyStack.

2 MyQueue

```
# A queue for project1 of CMPT435.

__author__ = "Tim Polizzi"
__email__ = "Timothy.Polizzi1@marist.edu"

from Assignments.AssignmentOne.MyLinkedList import MyLinkedList


class MyQueue(MyLinkedList):
    """A queue generated without the assistance of external
    libraries for CMPT435.

    A queue generated separate from any pre-existing libraries, for
     the purpose of learning how to use
    Queues for Allan Labouseur's algorithms class.
    """
    def __init__(self):
        """Initializes MyQueue"""
        super().__init__()
        self.inner_list = MyLinkedList()
        self.inner_list.tail = None

    def enqueue(self, to_enqueue: str):
        """Adds an item to MyQueue.

        Appends an item to the end of MyQueue.

        Args:
            to_enqueue: The item that is to be appended to MyQueue.
        """
        new_node = self.inner_list.TextNode(to_enqueue, None)
        if self.inner_list.is_empty():
            self.inner_list.head = new_node
            self.inner_list.tail = new_node
        else:
            self.inner_list.tail.next = new_node
            self.inner_list.tail = new_node

    def dequeue(self) -> str:
        """Removes an item from MyQueue.

        Removes the foremost item from MyQueue and returns it's
    value.

        Returns:
            The value of the item that was removed from MyQueue.
        """
        if self.inner_list.is_empty():
            raise Exception('List is empty')
        else:
            old_head_val = self.inner_list.head.val
            self.inner_list.head = self.inner_list.head.next
            return old_head_val
```

The implementation of MyQueue was done the same as with MyStack, which

was by using a LinkedList as an internal structure. This LinkedList only served as a place to hold the data, as both the enqueue and dequeue methods were taken care of through MyQueue.

2.1 Enqueue (21-35)

The method enqueue is intended to add a new item to MyQueue, which it does by appending a new TextNode to the tail of the internal LinkedList, and then setting that node to be the new tail of the LinkedList.

2.2 Dequeue (37-50)

The dequeue method of MyQueue removes the first item from MyQueue and returns it. To do this it just sets the head value to the next item past head in the internal LinkedList.