Assignment 02

Overview:
In this assignment you will be asked to implement a stack and use it solve to determine what words in the file are a palindrome. Unlike the stack implemented in class, you will implement the stack using a doubly-liked list. Therefore, there are 3 steps to this assignment:

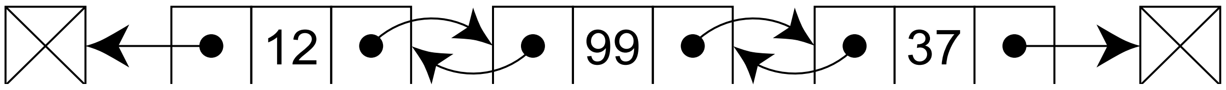
1. Implement a doubly-linked list
2. Implement a stack using the doubly-linked list from step 1
3. Use the stack from step 2 to determine if a word is a palindrome.

Step 01 – Doubly-Linked List
Each Node in a DoublyLinkedList has three properties:

- next :: Node
- prev :: Node
- data :: Any Type

Implement a doubly-linked list with the following methods. Your doubly-linked list class should have the following properties:

- head :: Node
- tail :: Node
- size :: Integer



__init__()	Construct an empty doubly-linked list object
isEmpty()	Returns true if doubly-linked list is empty
addFirst(element)	Add element to front of doubly-linked list
addLast(element)	Add element to end of doubly-linked list
removeFirst()	Remove first item in doubly-linked list
removeLast()	Remove last item in doubly-linked list
add(element, index :: int)	Add element to specified index, if invalid index display error message to user
remove(index :: int)	Remove node at specified index
__str__()	Forward traverse through the list and print all the values in each node (one per line)
__eq__()	Return true if two doubly-linked lists have the same content in the same order. Return false otherwise.
__add__(self, other : DoublyLinkedList)	Append *other* to the end of the doubly linked list *self*.
__len__()	Return length of doubly-linked list

Step 02 – Doubly-Linked List Based Stack

Implement a stack with the following methods. Your stack class should have the following properties:

- stk :: DoublyLinkedList
- size :: Integer

YOU MUST IMPLEMENT THE STACK WITH THE DOUBLY-LINKED LIST

__init__()	Construct stack object
isEmpty()	Returns true if stack is empty
push(element)	Add element to top of stack
pop()	Remove element from top of stack
peek()	Look at value at top of stack but do not remove
__str__()	Display the following information: "Size: # Top: element"
__len__()	Return size of stack
__add__(self, other : Stack)	Add *other* to the top of the stack *self*.

Step 03 – Determine Palindrome

In this assignment you will be asked to write a program that prompts the user for a word via keyboard and displays a message indicating if the word is a palindrome.

A palindrome is a word that is spelt the same was forward as it is backwards. In this assignment you must check if a word is a palindrome using the Stack data structure you developed!

- **Hint:** You only need to push half the word on to the stack
- Consider the cases of odd vs even length words.

Sample expected output for both positive and negative cases are show below.

Example Input 1:

```
Please enter a word: Radar
Radar is a palindrome!
```

Example Input 2:

```
Please enter a word: Boat
Boat is not a palindrome!
```