MINIA UNIVERSITY
FACULTY SCIENCE
Department of Computer Science
Data Structures Using Python

Exercises #7
Linked List

7.1 Create a class, named ***SinglyLinkedList***, which has a ***constructor*** that initializes its members, **_head** to ***None*** and **_size** to ***0***, and supports the following operations:

is_empty(), which returns True if the list is empty, otherwise returns False.
len(), which returns the number of elements in the list
add_first(), which adds an element at the head of the list.
add_last(), which adds an element at the end of the list.
remove_last(), which removes the element at the end of the list, and raises Empty exception if the list is empty.
remove_first(), which removes the element at the head of the list, and raises Empty exception if the list is empty.
display(), which displays the elements of the list. It displays the singly linked list 10->15->20 as [10, 15, 20]
contains(), which returns True if the list contains an item, otherwise returns False.
insert_before(), which adds an element before a given item, if exists.
insert_after(), which adds an element after a given item, if exists.
remove_item(), which removes a specific item, if exists.
reverse(), which reverses the elements of the list.

Note that the ***SinglyLinkedList*** class nodes are stored in class **_Node**, which is defined as follows:

```
class _Node:
    def __init__(self, element, next): # initialize node's fields
        self._element = element        # reference to user's element
        self._next = next              # reference to next node
```

7.2 Using the class ***SinglyLinkedList***, write a method ***ConcatLists()*** that concatenates two given linked lists, and returns the resulted list. Then, write a program that creates two linked lists, uses the method ***ConcatLists()*** to concatenate then then displays the elements of the new list.

7.3 Implement a ***LinkedStack*** class that inherits from the ***_DoublyLinkedBase*** class. Then, write a program that tests this class.

7.4 Implement a ***LinkedQueue*** class that inherits from the ***_DoublyLinkedBase*** class. Then, write a program that tests this class.

7.5 Repeat Exercises #6 using ***LinkedStack*** & ***LinkedQueue*** classes instead of ***ArrayStack & ArrayQueue Classes***.

7.6 Write a Python program that searches for a given element in a circularly linked list. If the element is present in the linked list, it prints "the element is found", otherwise, it prints "the element is not found".

7.7 Implement the class ***SinglyLinkedList*** using the ***collections.deque*** class, such that it supports the following operations:

is_empty(), which returns True if the list is empty, otherwise returns False.
len(), which returns the number of elements in the list
add_first(), which adds an element at the head of the list.
add_last(), which adds an element at the end of the list.
remove_last(), which removes the element at the end of the list.
remove_first(), which removes the element at the head of the list.
display(), which displays the elements of the list.
contains(), which returns True if the list contains an item, otherwise returns False.
remove_item(), which removes a specific item, if exists.
insert(), which adds an element at an arbitrary position
first(), which accesses the first element.
last(), which accesses the last element.
getElem(), which accesses an arbitrary element by index
modify(), which modifies an arbitrary element by index.
clear(), which clears all contents of the list.
reverse(), which reverses the elements of the list.