Practice 7

[Lecture 10-1] Arrays and Linked lists
[Lecture 10-2] Queues and Stacks

Seoul National University
Graduate School of Data Science

01. Exercise

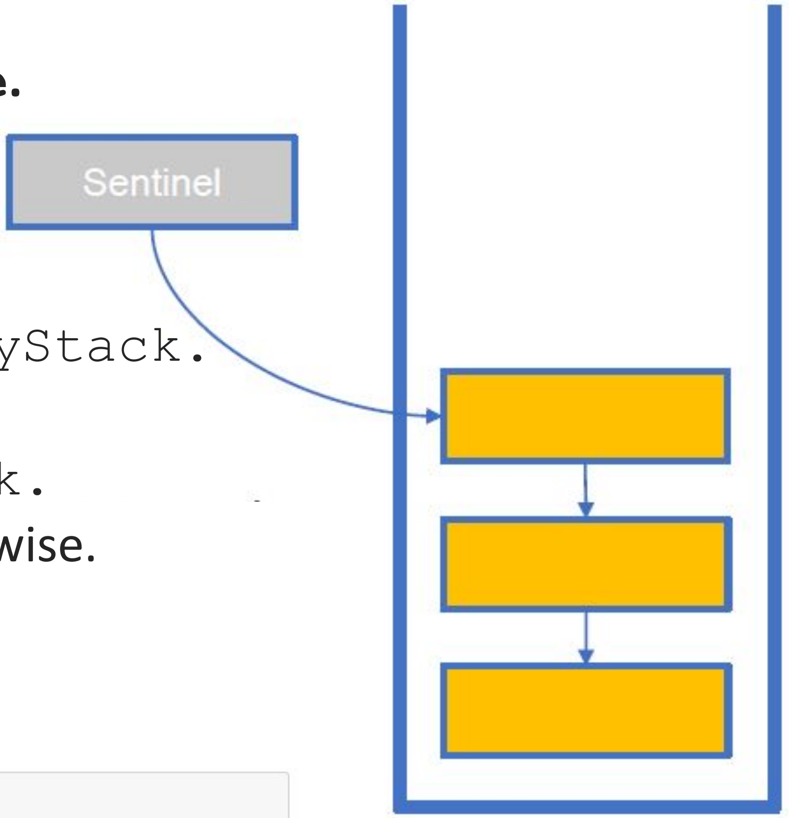
1. myStack

Implement a Stack with `myStack` using given class `LinkedNode`.

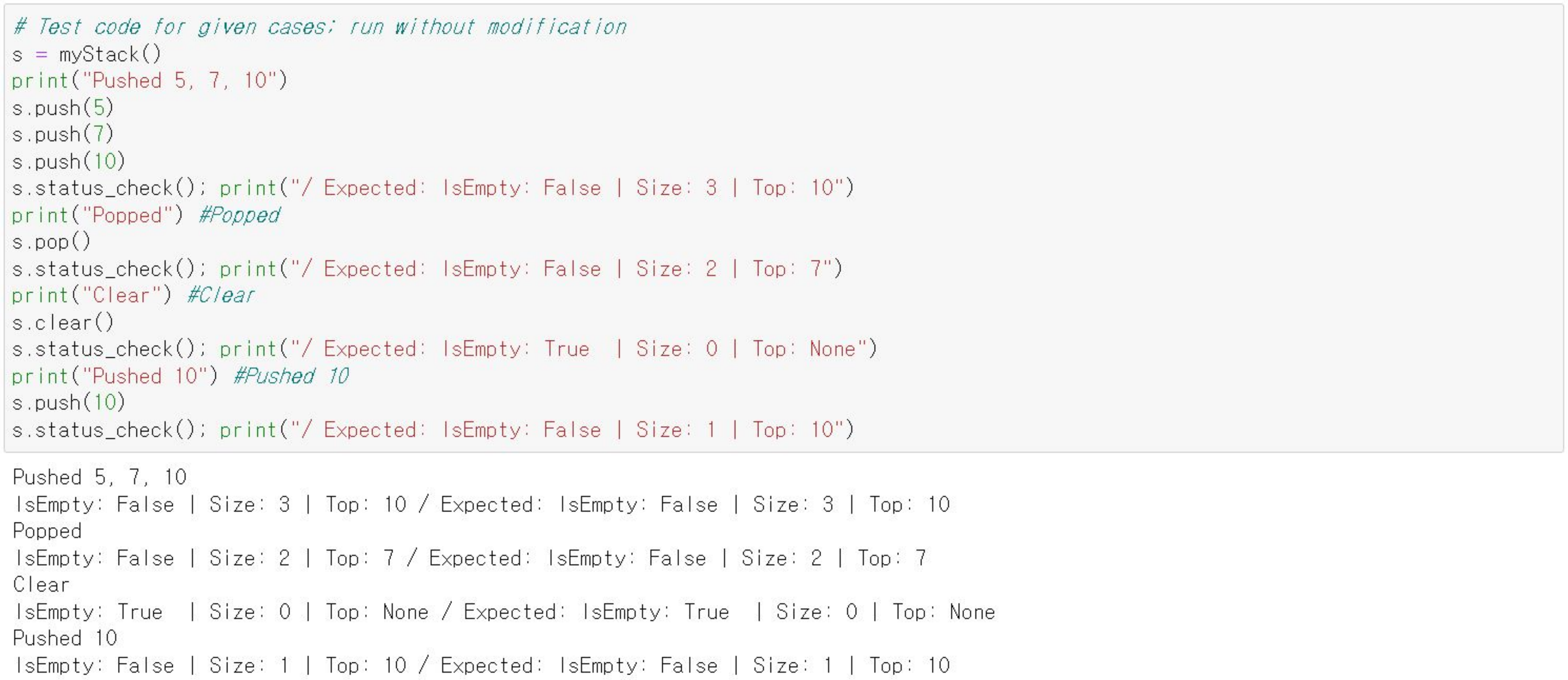
- Implement the following methods

1. `push(x)`: Add a `LinkedNode` that has val x to `myStack`.
2. `pop()`: Remove the most recently added `LinkedNode` from `myStack`.
3. `top()`: Return val of the most recently added `LinkedNode`.
4. `getSize()`: Return the number of `LinkedNode`s in `myStack`.
5. `isEmpty()`: Return True if `` `myStack` `` is empty, or False otherwise.
6. `clear()`: Remove all `` `LinkedNode` ``s.
7. `status_check()`: prints status of the stack.

```python
# Run without modification
class LinkedNode():
    def __init__(self,x):
        self.val = x
        self.next = None
```



1. myStack



```python
# Test code for given cases: run without modification
s = myStack()
print("Pushed 5, 7, 10")
s.push(5)
s.push(7)
s.push(10)
s.status_check(); print("/ Expected: IsEmpty: False | Size: 3 | Top: 10")
print("Popped") #Popped
s.pop()
s.status_check(); print("/ Expected: IsEmpty: False | Size: 2 | Top: 7")
print("Clear") #Clear
s.clear()
s.status_check(); print("/ Expected: IsEmpty: True  | Size: 0 | Top: None")
print("Pushed 10") #Pushed 10
s.push(10)
s.status_check(); print("/ Expected: IsEmpty: False | Size: 1 | Top: 10")
```

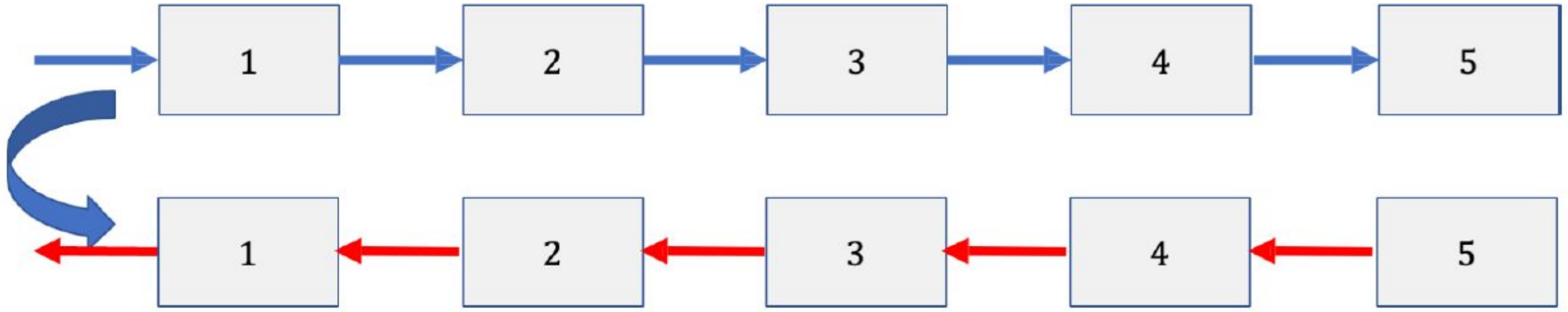
```
Pushed 5, 7, 10
IsEmpty: False | Size: 3 | Top: 10 / Expected: IsEmpty: False | Size: 3 | Top: 10
Popped
IsEmpty: False | Size: 2 | Top: 7 / Expected: IsEmpty: False | Size: 2 | Top: 7
Clear
IsEmpty: True  | Size: 0 | Top: None / Expected: IsEmpty: True  | Size: 0 | Top: None
Pushed 10
IsEmpty: False | Size: 1 | Top: 10 / Expected: IsEmpty: False | Size: 1 | Top: 10
```

2. Reverse SLL

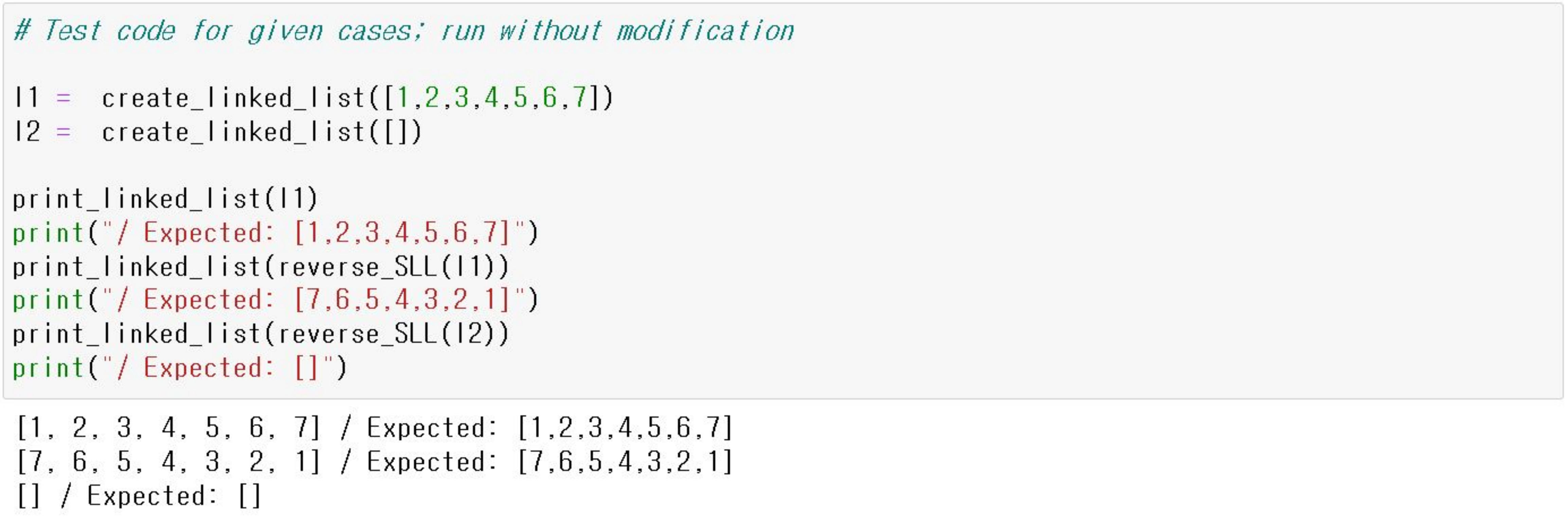
Implement functions `create_linked_list` and print `linked_list` using the predefined `LinkedNode` class. Then implement function `reverse_SLL` that takes in the head of a SLL and returns the head of a reversed SLL.

- Conditions

1. `create_linked_list` takes a Python list and returns the head of the created linked list
2. `print_linked_list` takes the head of a linked list and prints the values in it
3. Space complexity of `reverse_SLL` should be O(1).
   (Generating new linked lists or lists is not allowed)



2. Reverse SLL



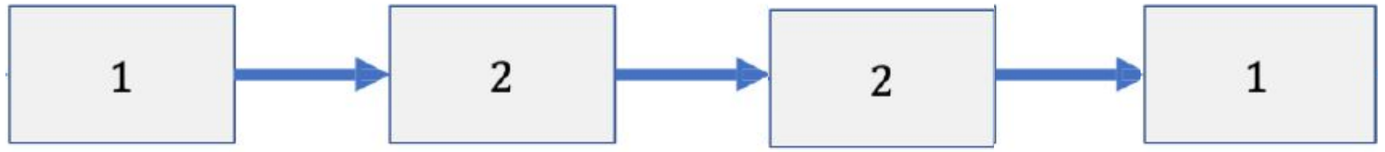
```python
# Test code for given cases; run without modification

l1 =  create_linked_list([1,2,3,4,5,6,7])
l2 =  create_linked_list([])

print_linked_list(l1)
print("/ Expected: [1,2,3,4,5,6,7]")
print_linked_list(reverse_SLL(l1))
print("/ Expected: [7,6,5,4,3,2,1]")
print_linked_list(reverse_SLL(l2))
print("/ Expected: []")
```

```
[1, 2, 3, 4, 5, 6, 7] / Expected: [1,2,3,4,5,6,7]
[7, 6, 5, 4, 3, 2, 1] / Expected: [7,6,5,4,3,2,1]
[] / Expected: []
```

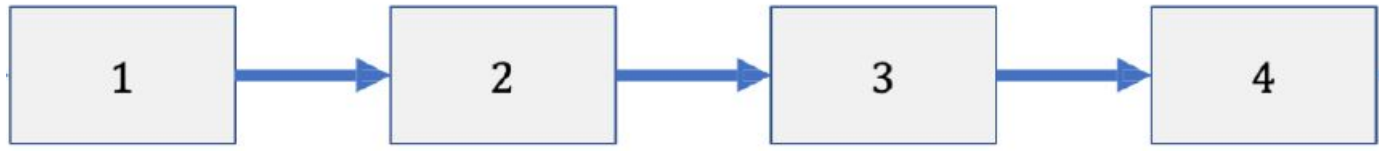
3. Palindrome SLL

Define function `isPalindrome` that takes the head of a SLL and returns whether it is a palindrome..

- Conditions
1. Return in boolean.
2. Try doing it without reversing the whole linked list!

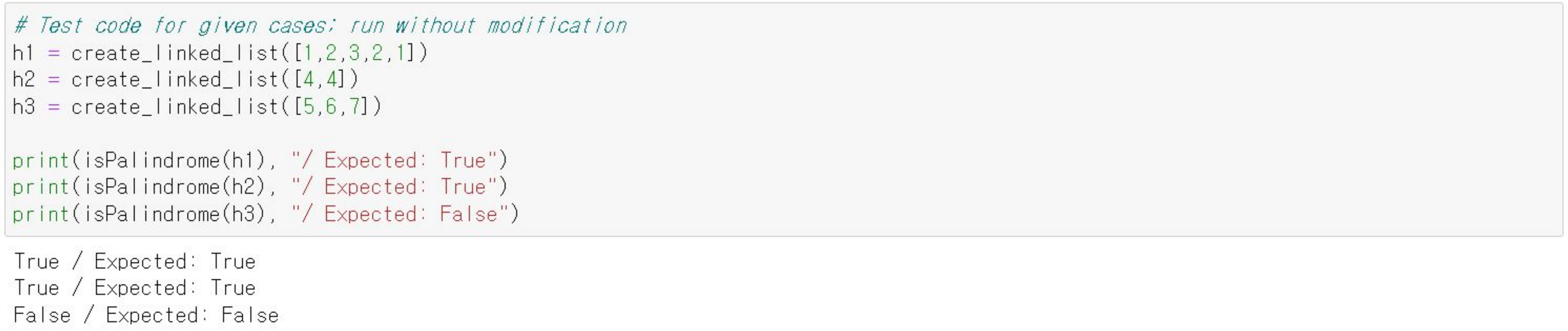


True



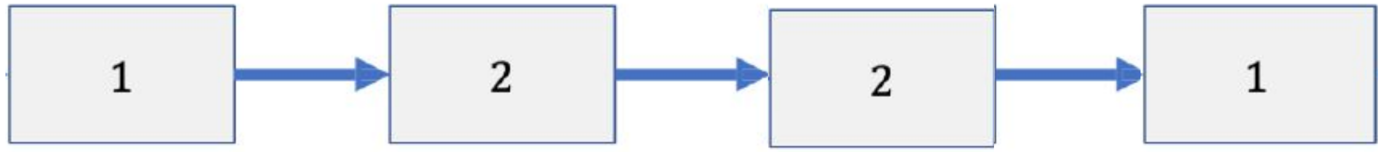
False

3. Palindrome SLL

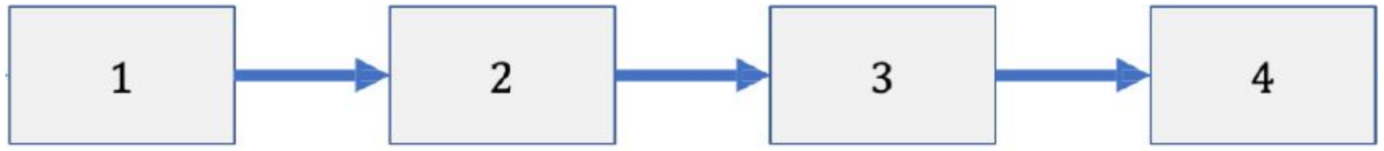


```
# Test code for given cases: run without modification
h1 = create_linked_list([1,2,3,2,1])
h2 = create_linked_list([4,4])
h3 = create_linked_list([5,6,7])

print(isPalindrome(h1), "/ Expected: True")
print(isPalindrome(h2), "/ Expected: True")
print(isPalindrome(h3), "/ Expected: False")
```

```
True / Expected: True
True / Expected: True
False / Expected: False
```



True



False

Breakout room guidelines

- 조를 짜신 분들은 빈 소회의실에 들어가서 자유롭게 실습하셔도 좋습니다.
- 실습 중에 질문이 있다면 본 줌 미팅에서 채팅 혹은 손들기 후 질문해도 괜찮습니다.
- 조를 아직 안 편성하셨거나 다른 분들과 토의하시고 싶은 분들 또한 소회의실에 접속하셔도 좋습니다.