Mod 4 Homework - Implementing a Turn Tracker for a Multiplayer Game

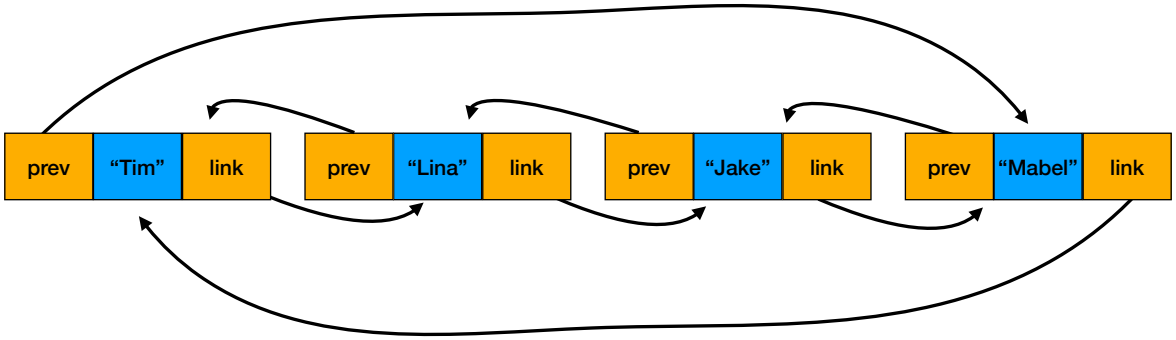
For this homework, we are going to implement an ADT for a multiplayer game, a **TurnTracker**. The objective of this assignment is to build our familiarity with doubly linked lists. After completing it, we should feel more comfortable using, extending, and modifying the doubly linked list data structure.

Problem Description

We are working on a turn tracker that would be used for a game similar to the multiplayer card game Uno. At the beginning of the game, each player is added to the turn tracker via the `addPlayer` method. After this is done, the initial turn order will follow the order in which players were added to the tracker. For each turn in the game, we need to will make a `nextPlayer` call to our turn tracker. This call will return the player that should play the next card.

However, during the game, players may play cards that modify the turn order, such as Skip and Reverse cards. When those cards are played, an associated method is going to be called on the turn tracker, `skipNextPlayer` and `reverseTurnOrder`, respectively.

Your turn tracker must be implemented with a **circular doubly linked list** data structure. This data structure is an extension to the doubly linked list that we have seen in class. A circular doubly linked list is just a doubly linked list where the `_head` and `_tail` nodes have links to each other rather than `None`. A turn tracker based on a circular double linked list holding the names of players is shown below.



Circular Doubly Linked List
Turn Tracker

Turn Tracker Behavior

Your `TurnTracker` class must implement the following methods:

- `addPlayer(player)`
  - adds a player to the turn tracker after the last player that was added to the tracker (i.e., this player should be added directly after the `_tail` node and then become the new `_tail`)
- `nextPlayer()`
  - returns the next player in the turn order based on the current game state
  - if called when there are no players in the turn tracker, should raise a `RuntimeError`
- `numberOfPlayers()`
  - returns the number of players in the turn tracker
- `skipNextPlayer()`
  - causes the next player in the turn order to be skipped
- `reverseTurnOrder()`

- reverses the current direction of the turn order

IMPORTANT:

- Every method implemented should have a docstring and $O(1)$ time complexity.
- In addition to the normal doubly linked list attributes (`_head` , `_tail` , and `_length`), it would be a good idea to have attributes such as:
  - `_nextPlayer` : a pointer that is moved around the list in order to determine the next player
  - `_reversed` : a Boolean that indicates if the tracker is in a reversed state
  - `_skipping` : a Boolean that indicates if the next player in the order should be skipped

Some Examples of Proper Behavior

Below are some examples of correct turn tracker behavior. (**Hint**: These would be good unit tests ;-) .)

Example 1

```python
tt = TurnTracker()
tt.addPlayer("Jake")
tt.addPlayer("Lina")
tt.addPlayer("Tim")

print(tt.nextPlayer())
print(tt.nextPlayer())
print(tt.nextPlayer())
print(tt.nextPlayer())
print(tt.numberOfPlayers())
```

Expected Output

```
Jake
Lina
Tim
Jake
3
```

Example 2

```python
tt = TurnTracker()
tt.addPlayer("Jake")
tt.addPlayer("Lina")
tt.addPlayer("Tim")

print(tt.nextPlayer())
print(tt.nextPlayer())
tt.reverseTurnOrder()  # Lina plays reverse card
print(tt.nextPlayer())
print(tt.nextPlayer())
print(tt.nextPlayer())
```

Expected Output

```
Jake
Lina
Jake
Tim
Lina
```

Example 3

```python
tt = TurnTracker()
tt.addPlayer("Jake")
tt.addPlayer("Lina")
tt.addPlayer("Tim")

print(tt.nextPlayer())
print(tt.nextPlayer())
print(tt.nextPlayer())
tt.skipNextPlayer()    # Tim plays Skip card
print(tt.nextPlayer())
tt.skipNextPlayer()    # Lina plays Skip card
print(tt.nextPlayer())
print(tt.nextPlayer())
```

Expected Output

```
Jake
Lina
Tim
Lina
Jake
Lina
```

Example 4

```python
tt = TurnTracker()
tt.addPlayer("Jake")
tt.addPlayer("Lina")
tt.addPlayer("Tim")

print(tt.nextPlayer())
print(tt.nextPlayer())
print(tt.nextPlayer())
tt.skipNextPlayer()    # Tim plays Skip card
print(tt.nextPlayer())
tt.reverseTurnOrder()  # Lina plays Reverse card
print(tt.nextPlayer())
tt.skipNextPlayer()    # Jake plays Skip card
print(tt.nextPlayer())
print(tt.nextPlayer())
```

Expected Output

```
Jake
Lina
Tim
Lina
Jake
Lina
Jake
```

Imports

No imports allowed on this assignment, with the following exceptions:

- Any modules you have written yourself

- For testing only (do not use these for functionality in any other classes/algorithms):
  - unittest
  - random

Submission Requirements

Students must submit **individually** by the due date (Tuesday, October 3rd at 11:59 pm EST) to receive credit.

At a minimum, you must submit the following files:

- `turnTracker.py`
  - This file should include implementions for all of the necessary turn tracker methods.
- `testTurnTracker.py`
  - This file should include unit tests that test for proper turn tracker behavior. You should have at least one unit test for each method implemented.

Additionally, you must include any other files necessary for your code to run, such as modules containing data structures you wrote yourself.