LAB04: Snap!
CSE2050 Fall 2017
Instructor: Jeffrey Meunier & Wei Wei
TA: Jenny Blessing, Param Bidja, Yamuna Rajan & Zigeng Wang

1. Introduction

In this exercise, we will use a data structure known as a LinkedDeque (deque implemented using linked list) to implement the card game Snap!

2. Objectives

The purpose of this assignment is to help you:

- Sharpen your knowledge on basic data structures through an application
- Refine your ability to translate real-world scenarios into code

***Note**: Before you start, if you are not familiar with LinkedLists or D*eques *you are recommended to review the sample codes we covered in lecture first.*

3. Background

3.1. Snap Card Game

Snap is an easy-to-learn card game. Our version of Snap makes the game even more straight forward. The basic flow of our version of Snap is as follows:

- Step 1: Divide the deck into 2 halves (1 half per player; deck does not include Jokers)
- Step 2: While neither player has an empty deck, the two players take turns throwing a card into a public deck (which starts off as an empty deck). If a card is thrown that has the same rank as a previous card in the public deck, the player who threw that card takes the cards in the public deck up until (and including) the card of the same rank. **Note: the cards thrown by each player can be from either the top or the bottom of their deck. We will determine whether the top or bottom card is thrown using randomness.**

Consider this example: suppose player A has the cards A, 2, Q, J, K, 5, Q, player B has the cards 3,2, Q, 7, 4, and the public deck has the card 7, 5, 4, 1, 8 (this example is not with 52 cards, it is simplified for the sake of explanation). Player A randomly decides to throw the card that's on the bottom of his/her deck, a Queen, which has not appeared thus far in the public deck (so the new public deck is 7, 5, 4, 1, 8, Q). Then player B throws a 4 which has appeared in the public deck (now the public deck is 7, 5, 4, 1, 8, Q, 4), thus player B takes the public deck up until and including the 4 and adds it to the bottom of his/her deck (4, 1, 8, Q, 4 – in exactly that order). After this one cycle of turns, player A has the cards A, 2, Q, J, K, 5, player B has the cards 3, 2, Q, 7, 4, 1, 8, Q, 4**,** and the public deck has 7, 5**.**

4. Assignment

In the skeleton zip file, you can find a skeleton code. All you need to do is to complete the skeleton code based on the following instructions and submit the to Mimir.

4.1 display

In the LinkedList class, you will find an empty display() function. This function should loop through all elements of the LinkedList and display them. Notice an empty print statement is included at the end of this function, that is for a new line (**do not delete**). **Note: display will not be graded, it is meant to help you for the next section of this lab.**

An example test is:

Input:

```
n = LinkedList()
n.addfirst('1729')
n.addlast('2050')
n.display();
```

Output:

```
1729 2050
```

4.1 playgame(seed)

This function simulates the snap game. The parameter 'seed' is used to make the randomness predictable for testing reasons.

1. The first step in simulating the game is dividing the deck into 2 halves. To do this, simply split the deck list into 2 lists - one for player 1 and one for player 2. **Player A gets the first half (i.e. the first 26 cards in the list of cards) and player B gets the second half (i.e. the last 26 cards in the list of cards).** After splitting the deck into 2 lists, create and populate a LinkedDeque() for each player. The public deck should be initialized to an empty list.

The example with seed 305 looks like this after step 1:

Public deck: []

A's deck: ['6', 'A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10', '5']

B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J', 'Q']

2. The next step is simulating the cycles of the game. The game continues as long as both players have cards in their deck.

    1. The first step in simulating the game is having player A throw a random card. Suppose player A's deck is held in lA, we'll define randomness as follows:

```
rand = random.random()
        if rand > 0.5:
            card = lA.removelast()
        else:
            card = lA.removefirst()
```

    Make sure you use this definition of random, as we are testing based on this.

    2. After player A throws a card into the public deck, you need to check if that card exists in the public deck. If so, **add all cards in the public deck from the card until the end of the deck, including the card itself to the bottom of player A's deck.** If not then simply add the card player A threw into the public deck.

        - **For the playgame(305) example, the decks look like this after the 1st time Player A:**
            - Public deck: ['5']
            - A's deck: ['6', 'A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10']
            - B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J', 'Q']

    3. Next, you must check to see if player A has an empty deck. If so, the game is over and Player B has won. If not, repeat steps 1, 2, and 3 for player B.
        - **For the playgame(305) example, the decks look like this after the 1st time Player B:**
            - Public deck: ['5', 'Q']
            - A's deck: ['6', 'A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10']
            - B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J']

3. Once the game has ended, **return a tuple with the first part being either "A wins" or "B wins" and the second part being the winning player's final deck as a list** (hint: use the alldata() function in the LinkedDeque class). Here are some examples:

```
print(playgame(305))
('A wins', ['2', 'A', '8', '7', '8', 'A', '4', '5', 'K', '4',
'6', 'Q', '8', 'K', '7', '3', 'K', '2', '3', '0', '5', '2', 'J',
'4', '9', '8', '6', '0', 'Q', '7', '2', '6', '5', '9', 'Q', 'J',
'7', '3', 'J', 'A', '0', '4', '3', '9', '9'])
```

After splitting the deck, the decks look like:

Public deck: []

A's deck: ['6', 'A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10', '5']

B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J', 'Q']

After the 1st time Player A throws a card, the decks look like:

Public deck: ['5']

A's deck: ['6', 'A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10']

B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J', 'Q']

Then, after Player B throws his/her first card:

Public deck: ['5', 'Q']

A's deck: ['6', 'A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10']

B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J']

Then after Player A's second turn:

Public deck: ['5', 'Q', '6']

A's deck: ['A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10']

B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q', 'J']

Then after Player B's second turn:

Public deck: ['5', 'Q', '6', 'J']

A's deck: ['A', 'A', '7', 'Q', '8', '10', '4', '9', '9', '6', 'K', '3', '10', '3', 'Q', '7', '3', '2', '3', '10', '2', '5', '7', '10']

B's deck: ['A', '8', '8', 'J', '9', '2', '6', '4', '4', '9', 'K', '7', 'J', '5', '5', 'J', 'A', 'K', '8', '6', '4', '2', 'K', 'Q']

Here are some additional examples with the final result shown:

```
print(playgame(400))
('A wins', ['2', '7', '9', '0', '5', '2', '4', '6', '5', '9',
'4', '7', '8', '5', 'A', 'Q', '0', 'Q', '3', 'J', 'K', '7', '3',
'6', 'K', '8', 'Q', '7', '0', 'A', '3', 'J', '9', '4', 'A', '2',
'K', '8', 'J', '2', '8', '9', 'K', '5', '4', '6', 'J', 'A',
'6'])

print(playgame(1300))
('A wins', ['K', '4', '3', '4', '2', '4', '0', '2', 'Q', '5',
'6', '3', 'K', '2', '9', '9', '0', 'J', '7', '6', '8', '5', 'A',
'6', '0', 'A', '5', 'A', 'Q', '4', '8', 'Q', '7', '9', 'J', '2',
'8', '0', 'Q', '8', '3', '5', '7', '9', '6', 'K', 'J'])
```

5. Submit your work to Mimir

Submit snap.py to Mimir after you complete your code. The Mimir will automatically grade your submission based on different unit tests. You can submit your code to Mimir up to **30 times** to refresh your existing score before the submission deadline.

6. Due date

This lab assignment is worth 2 points in the final grade. It will be due by 11:59pm on Wed. Oct 4th 2017. A penalty of **10% per day** will be deducted from your grade, starting at 12:00am.

7. Getting help

Start your project early, because you will probably not be able to get timely help in the last few hours before the assignment is due.

- Go to the office hours of instructors and TAs.
    - Prof. Wei Wei: Mon. 2:30 - 3:15pm, Wed. 2:30 - 3:15pm, Thurs. 2:30 - 3:15pm, Fri. 2:30 - 3:15pm @ITE258
    - Jenny Blessing: Fri. 12pm - 2pm @ITE140
    - Param Bidja: Tues. 2pm - 3pm @ITE140
    - Yamuna Rajan: Tues. 11am - 12pm, Wed. 9:30am – 10:30am @ITE140
    - Zigeng Wang: Mon. 3pm - 4pm @ITE140
- Post your questions on Piazza. TAs and many of your classmates may answer your questions.
- Search the answer of your unknown questions on the Internet. Many questions asked by you might have been asked and answered many times online already.