Assignment 2

(Due: Nov 26, 2019) 11:59PM [NO EXTENSIONS WILL BE GRANTED]

Total points: 100

Introduction

The goal of this assignment is to understand and implement a circular doubly linked list. Be sure to carefully read each of the problems, as every detail will play a role in the solution. The assignment is worth a total of 100 points.

This assignment is to be completed **individually**. You are NOT allowed to share nor acquire source code from other students in the class, current or previous. You must start the assignment early in order to get proper help from the TAs and the instructor. You can also post general questions or ask for clarifications on Piazza, however, **do not post your own source code.**

Please read the entire assignment carefully, before starting

Methodology

You are required to create a source file `assignment2.py` that will contain the solutions to the question below. For each question you will create a method using the provided defintions. You may add any number of helper methods.

PLEASE USE the method definitions exactly as given for each question. If you do not, then the autograder on Gradescope will not be able to recognize your answer and will give you a zero.

Implementing a Circular Doubly Linked List

You will design and implement a class for a customized Circular Doubly-Linked List named CDLL which contains at least the following public methods: `insert()`, `print_current()`, `go_next()`, `go_prev()`, `go_first()`, `go_last()`, and `skip()` (refer to the comments below for more details). Each node in the list stores two strings, corresponding to the time and text of a Tweet (from Twitter). Your classes must be implemented in the `assignment2.py` file.

```
class CDLLNode:
    def __init__(self, time="", tweet="", next_node=None, prev_node=None):
    self.time: str = time
    self.tweet: str = tweet
    self.next_node: CDLLNode = next_node
    self.prev_node: CDLLNode = prev_node
```

```
class CDLL:
    def __init__(self):
        self.head: CDLLNode = None
        self.current: CDLLNode = None
        self.numnodes: int = 0
    ...
    # makes an insertion based on the 'current' node
    def insert(self,time: str, tweet: str):

    # moves 'current' pointer to the next node (circularly)
    def go_next(self):

    # moves 'current' pointer to the previous node (circularly)
    def go_prev(self):

    # moves 'current' pointer to the head (the first node)
    def go_first(self):

    # moves 'current' pointer to the last node
    def go_last(self):

    # moves 'current' pointer n elements ahead (circularly)
    def skip(self,n:int):

    # prints the contents of the 'current' node
    # prints the time, then the tweet (each with a newline following)
    def print_current(self):
```

Using your class in a Tweet Reader application

You will develop a program that will open and read the contents of an input **test file**. The file name will be provided **as a command line argument**. This means that your code will need a `main()` function. In addition to this, you need to use the method we've been using in labs for making sure that the `main` runs, by including this code at the bottom of your file:

```
if __name__ == "__main__":
    main()
```

Each **test file** is a text file containing tweets of a news agency. For example, `bbchealth.txt` is related to BBC health news. Each line contains tweets following the format: `tweet id|date` and `time|tweet`. You can assume the separator between fields on each line is always `'|'`.

Here is an example tweet from the dataset:

```
515169518445146112|Thu Sep 25 16:02:39 +0000 2014|American #Ebola patient
Dr. Rick Sacra has been released from the hospital
```

In this example, `515169518445146112` is the tweet id, the `date` value is `Thu Sep 25`, the `time` value is (only): `16:02:39` (It's in military time), and the `tweet` itself is: `American #Ebola patient Dr. Rick Sacra has been released from the hospital`

Example: If this tweet were the current tweet, and the user asked to 'print the current tweet', we would print:

```
16:02:39
American #Ebola patient Dr. Rick Sacra has been released from the hospital
```

All test files were downloaded from the course web-page as [Twitter Dataset]

While reading the contents of the input file line-by-line, your program will be inserting each tweet into a circular doubly linked list in chronological order.

> The first tweet should be closest to the time 00:00:00, and the last tweet should be closest to the time 23:59:59

While reading the contents of the input file line-by-line, your program will be inserting each tweet into a circular doubly linked list in chronological order.

After reading all tweets, your program will print to the `stdout` the earliest tweet and then enter a loop waiting for user commands. The user can interact with your program using one of the following commands:

- `n` : prints the next tweet (chronologically) to the stdout
- `p` : prints the previous tweet (chronologically) to the stdout
- `f` : prints the first tweet (closest to 00:00:00) to the stdout
- `l` : prints the last tweet (closest to 23:59:59) to the stdout
- `num` : prints the number of tweets stored in the list
- `<number>` : skips tweets circularly and prints the current to the stdout
- `s <word>` : searches for the next occurrence of the substring word in the following tweets (search is

case insensitive and performs a circular traversal in the list)

    - If the word was found, print the contents of the tweet that had the word
    - If the word was not in your list, print "Word not found"

- `q` : quits the program

Do not print any words in your input call. Just use `input()` . If you *do* print words in the input call, i.e.,: `input("Please enter a command")` , the autograder will *NOT BE ABLE TO* account for it and you won't get any points for this section.

Point Distribution

The sections below show the distribution of points for the assignment:

Linked List Methods	30 Points Total
`insert()`	10 points
`go_next()`	3 points
`go_prev()`	3 points
`go_first()`	3 points
`go_last()`	3 points
`skip()`	5 points
`print_current()`	3 points

Tweet Reader Input Functionality	50 Points Total
`n` (update `current` as the next tweet, print contents)	4 points
`p` (update `current` as the previous tweet, print contents)	4 points
`f` (update `current` as the first tweet, print contents)	4 points
`l` (update `current` as the last tweet, print contents)	4 points
`num` (prints the number of tweets in the list)	8 points
`<number>` (update `current` as the tweet `<number>` tweets away, print contents)	10 points
`s <word>` (search for a word from the tweets in your list)	14 points
`q` (quit the program)	2 points

Additional Tests	20 Points Total
Test for your tweets being in chronological order	10 points
Test for two random tweets to be where they should be	10 points

NOTE: we can test your submitted code with any .txt file in the Dataset, not just BBCHealth.txt given in the example above

SUBMISSION

You will submit ONE file called `assignment2.py` to Gradescope for this lab.

`assignment2.py` should contain your entire source code for this assignment.

Please provide meaningful comments and use proper coding style and indentation. There is no need to upload additional files. Feel free to create additional private/public methods, but you can't change/add data members.

Your program will be automatically graded. For each of the aforementioned tests you either pass the test case (full points) or not (zero points).

Remember, we can test with any .txt file in the Dataset.

Students caught cheating or plagiarizing will receive no credit. Additional actions, including a failing grade in the class or referring the case for disciplinary action, may also be taken.

Late submissions will receive a ZERO.

NOTE: Gradescope allows me to compare all submissions with each other and see how similar they are. If I find submissions which are too similar to each others, all the similar looking assignments will receive a ZERO. Disciplinary action might be taken depending upon the severity of the issue.

Your Gradescope submissions will not show how many points you have until after the due date has passed. It is on you to follow the requirements for each section, and to come up with your own thorough test cases that make sure that each part is complete and correct. You are allowed *unlimited* resubmissions until the due date.
