CS2515: Algorithms and Data Structures I

Continuous Assessment 1

This lab is part of the formal continuous assessment for CS2515. It focuses on the implementation and use of the doubly linked list. It will count for up to 10% of the total marks available for CS2515.

Since this is part of the formal assessment, the University's rules on plagiarism apply. You should not copy anyone else's code, you should not download code obtained from any website, you should not get anybody else to write the code for you, and do not try to submit code that may have been submitted for similar assignments on previous years for this module. The work you submit must be your own. However, you are free to reuse any code that has been delivered this year in CS2515 lectures or labs.

The PyFlix Movie Library

1. Design and implement a class `Movie`, with fields `title`, `director`, `cast`, `length`, and `rating` (and you could have other fields for e.g. the category, the video file, the movie poster image, the the release date, etc., but we don't need them in this exercise). Your Movie class should have the following methods: `__init__(...)`, `__str__()`, and `get_info()`. The `init(...)` method requires a title, director, cast list as a string, and length as an integer number of minutes. The `rating` field is meant for the user's personal rating, and so should be -1 on creation to indicate no rating given. The `__str__()` method should return a short string containing title and director. The `get_info()` method should return a longer string containing info on all fields, with the name of each field listed before its content.

2. Design and implement a class `PyFlix`, which is essentially a list of movie files (Movie instances). The should offer the following methods (in addition to `__init__(...)`):

   - `__str__()`, which should return a single string containing each movie in the list on a separate line, where each movie is represented by its title, director. The currently selected movie, if any, should be preceded by '-->'.
   - `add_movie(movie)`, which should add a new movie to the end of the list
   - `get_current()`, which should return the currently selected movie
   - `next_movie()`, which should change the current selection to the next one
   - `prev_movie()`, which should change the current selection to the previous one
   - `reset()`, which should reset the curent selection to the list head
   - `rate()`, which gets a suitable integer rating from the command line and assigns it to the current movie
   - `info()`, which gets a string consisting of the full details of the current movie, including any rating, and prints it to the screen
   - `remove_current()`, which should remove the current movie from the list
   - `length()`, which should report the number of movies in the library

   The class `PyFlix` must be implemented as a doubly-linked list. It is easiest to implement PyFlix to deal directly with a sequence of DLLNodes (i.e. implement all the required functionality of doubly linked lists directly into the PyFlix class - you don't create a separate standalone DoublyLinkedList).

   For this exercise, for invalid input, if we are obliged to return an item, we will return `None`, and otherwise we will do nothing. If you delete the currently selected movie, make whetever came after it the new current movie; if that goes off the end of the list, reset the current movie to be the head.

3. Test your class using the following sequence of operations:

   i. create an empty PyFlix list
   ii. create the movie ("El Camino", "Vince Gilligan", "Aaron Paul", 122) and add it to the list
   iii. create the movie ("Joker", "Todd Phillips", "Joaquin Phoenix", 122) and add it to the list
   iv. create the movie ("Midsommar", "An Aster", "Florence Pugh", 138) and add it to the list

v. display the list to the screen
vi. set the current movie to be the next one (i.e. first in list)
vii. display the info for the current movie
viii. move current to the next movie
ix. report the current movie
x. ask the user to rate the current movie
xi. move the current pointer to the previous movie
xii. delete the current movie
xiii. display the full list to the screen
xiv. display the info for the current movie
xv. create the movie ("Hustlers", "Lorene Scafaria", "Constance Wu, Jennifer Lopez", 110) and add it to the list
xvi. move current pointer to the next movie
xvii. move current pointer to the next movie
xviii. display the info for the current movie
xix. display the full list to the screen

which should give you output something like:

```
$$$ PyFlix Library:
(El Camino, Vince Gilligan)
(Joker, Todd Phillips)
(Midsommar, An Aster)
$$$

Info: Title: El Camino; Director: Vince Gilligan; Cast: Aaron Paul; Length: 122 minutes; Rating: None

Current movie: Joker, Todd Phillips

Current movie is Joker, Todd Phillips Enter your rating [0 to 4]: 3
Your rating was 3 $$$ PyFlix Library:
--> (Joker, Todd Phillips)
(Midsommar, An Aster)
$$$

Current movie: Hustlers, Lorene Scafaria, comedy

Info: Title: Joker; Director: Todd Phillips; Cast: Joaquin Phoenix; Length: 122 minutes; Rating: 3
Info: Title: Hustlers; Director: Lorene Scafaria; Cast: Cosntance Wu, Jennifer Lopez; Length: 122
minutes

$$$ PyFlix Library:
(Joker, Todd Phillips)
(Midsommar, An Aster)
$$$
```

This is just a simple example so that you can check that you are producing the right sort of output. You are strongly advised to test your own code using more extensive test methods than this. We will be using different methods to test your submitted software.

4. Add the following method to your `PyFlix` class:

   - `search(word)`, which moves the current movie pointer to the next movie in the list containing the input word as a substring anywhere in any of its fields; wrap the search round to the start again if you need to, and if no movie matches the input, the current pointer should be pointing to whatver it was pointing to before you started the search. After the search, if you found a movie, print its info, and if you didn't find a match, print "no matching movie";

Submission instructions

To submit, send an email to *k.brown@cs.ucc.ie* with your python file as an attachment. The title of your email should start CS2515 CA1. If you are emailing from a personal account, you must include your ID number in the text of the email.

The submission deadline is listed on the Problem Classes page.

- Do ***not*** be tempted to leave this until next week before making progress on the assignment. Evey year, people think there is plenty of time, but then run out of time before they can get it completed. You will require a lot of time to test, debug and re-test.
- You must submit a single file, which must be named `pyflix.py`. This file must contain your implementation of the Movie class, the PyFlixclass, any classes you need for the Doubly Linked List, and any test methods you used.
- The file must not contain tabs - use spaces for the Python indentation. Make sure you test the implementation on the machines in the lab before submitting (and don't edit the files in any way after your last successful test)
- Make sure that you compile and test your code before submitting, and that you submit exactly the version that you tested.