ASHESI UNIVERSITY

CS222: Data Structures and Algorithms
Summer Semester (May - Aug 2024)
Assignment 2 (Linked Lists)

STATEMENT ABOUT ACADEMIC HONESTY AND INTEGRITY

Academic honesty and integrity are very important at Ashesi and central to the achievement of our mission: To train a new generation of ethical and entrepreneurial leaders in Africa to cultivate within our students the critical thinking skills, concern for others, and the courage it will take to transform a continent. As this mission is our moral campus, we recommend you take it seriously in this course without any exceptions at all.

Ashesi therefore does not condone any form of academic dishonesty, including plagiarism and cheating on tests and assessments, amongst other such practices. Ashesi requires students to always do their own assignments and to produce their own academic work unless given a group assignment.

As stated in Ashesi’s student handbook, Section 7.4:

"Academic dishonesty includes plagiarism, unauthorized exchange of information or use of material during an examination, unauthorized transfer of information or completed work among students, use of the same paper in more than one course, unauthorized collaboration on assignments, and other unethical behaviour. Disciplinary action will be taken against perpetrators of academic dishonesty."

All forms of academic dishonesty are viewed as misconduct under Ashesi Student Rules and Regulations. Students who make themselves guilty of academic dishonesty will be brought before the Ashesi Judicial Committee and such lack of academic integrity will have serious consequences for your academic records.

INSTRUCTIONS:

1. This is an individual assignment. So, every student must independently work and submit.
2. This lab assignment contains *two* questions. You are required to solve both.
3. Each question carries 50 points. So, the total assignment will be evaluated for 100 points.
4. Your solution programs should contain comments explaining the program statements that involve important computations.
5. Alongside your program file, you are required to show the execution of your programs as a screen-recorded video for each task (max 8 min).
6. You need to upload the following to the submission portal.
   - ✓ your original '.java' files (by the deadline)
   - ✓ a screen-recorded video of the execution of your programs (by the deadline)
   - ✓ a document (pdf) containing answers to the questions given in class (after the deadline)
7. **Grading criteria**: points for each question are awarded as follows.
   - ✓ Solution program: 15 points
   - ✓ Comments on program statements: 5 points
   - ✓ Correct execution: 20 points
   - ✓ Test cases in the screen-recorded video demonstrating the correctness of the implementations: 5 points
   - ✓ Answering the in-class questions: 5 points (Please note that this may impact the points given for the other three elements above)

8. The deadline for submission is **Friday, 5th July 2024, 11.55 PM.**

9. Please refer to the course outline/syllabus document for the late submission policy.

PROBLEM 1: Browser History Management

The objective of this task is to strengthen your understanding of Doubly Linked Lists by implementing a browser history management system in Java. You will create a doubly linked list to manage the history of web pages visited in a browser, enabling operations such as adding new pages, removing pages, and displaying the history in both forward and backward order.

You are tasked to implement a Browser History Management System in Java. Your solution consists of three major parts.

Part 1: Implementing the HistoryNode Class

- Define a class HistoryNode with the following attributes:
  - url: The URL of the visited page.
  - timestamp: The time when the page was visited.
  - prev: Pointer to the previous node (initially null).
  - next: Pointer to the next node (initially null).

Part 2: Implementing the BrowserHistory Class

- Define a class BrowserHistory using a doubly linked list with the following methods:
  - public BrowserHistory(): Constructor to initialize the doubly linked list with head and tail set to null.
  - public void addPage(String url, String timestamp): Add a new page to the end of the history.
  - public void removePage(String timestamp): Remove a page by its timestamp.
  - public void displayHistoryForward(): Display the browsing history from the oldest to the newest.
  - public void displayHistoryBackward(): Display the browsing history from the newest to the oldest.

Part 3: Implement the Browser History Management Operations

- Create a BrowserHistory instance.
- Perform the following operations:
  - Add some pages to the history.

    For example, "www.example1.com" at 10:00 AM, "www.example2.com" at 10:05 AM, "www.example3.com" at 10:10 AM etc.
  - Remove the page visited at 10:05 AM.
  - Display the browsing history in forward order.
  - Display the browsing history in backward order.

Bonus: Implement a method to save and load the browser history to/from a file (up to 10 extra points)

PROBLEM 2: Music Playlist Manager

The objective of this task is to strengthen your understanding of singly linked lists, doubly linked lists, and circularly linked lists and their essential operations like insertion, deletion, and traversal. You will develop skills to apply linked lists to solve such real-world problems by writing modular and efficient code.

You are tasked to develop a music playlist management application using various linked list implementations (singly, doubly, and circular). The application should allow users to create, modify, and interact with playlists of songs.

You will create three different kinds of lists as detailed below. Please note that these sub-tasks are progressive, which means you do them in the same order building one based on the other.

1. **Singly Linked List (Basic):**
    - Implement a Song class with attributes like title, artist, and duration.
    - Create a **BasicPlaylist** class using a **singly linked list** to store Song objects.
    - Implement methods to:
        - Add a song to the end of the playlist.
        - Add a song at a specific position.
        - Remove a song by title or position.
        - Display the playlist in order.
        - Calculate the total duration of the playlist.

   Note: No specific method headers/signatures will be given. So, you will define them appropriately based on the requirement.

   Reflective question: What Java interface or class of the 'collections framework' will allow you to create the BasicPlaylist? Can you try writing this program using that class or interface?

2. **Doubly Linked List (Enhanced Navigation):**
    - Upgrade the Playlist class to use a **doubly linked list**. That means you write a separate class named **EnhancedPlaylist**.
    - Alongside the basic methods as in the basic singly linked list playlist you created above, add methods for:
        - Playing the next song.
        - Playing the previous song.
        - Shuffling the playlist (randomly rearranging songs).

   Note: Again, no specific method headers/signatures will be given. So, you will define them appropriately based on the requirement.

Reflective question: What Java interface or class of the 'collections framework' will allow you to create the EnhancedPlaylist? Can you try writing this program using that class or interface?

3. **Circularly Linked List (Continuous Play):**
    - Modify the EnhancedPlaylist class to use a **circularly linked list**. That means you write a separate class named **FullyFunctionalPlaylist**.
    - Implement continuous play mode where, after the last song, the playlist starts again from the beginning.
    - Add an option to toggle between continuous play and normal play.

Additional Features (Optional and ungraded, for improving your programming skills):

- Implement a search function to find songs by title or artist.
- Allow users to save and load playlists from files.
- Play the real audio songs on your computer when you invoke 'play' methods and stop them by invoking a 'stop' method.
- Create a GUI (Graphical User Interface) for the playlist manager considering the entire functionality.