ASHESI UNIVERSITY

CS222: Data Structures and Algorithms
Fall Semester (Aug - Dec 2023)
Lab Exercise 3 (Doubly Linked List)

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 of them.
3. Each question carries 50 Marks and thus the total assignment will be evaluated for 100 Marks. However, the first question has a weightage of 60% and the 2nd question 40%.
4. You are required to show the execution of your program as a screen-recorded video (max 5 min) and may have to answer oral questions to prove your knowledge and understanding of the concepts.
5. Your solution program should contain comments explaining statements that involve important computations.
6. You need to upload your original '.java' file, screen-recorded video, and a document (pdf/Word) containing an answer to the oral question given in the class.
7. Marks for each question are awarded as follows.
   Submission: 10%
   Execution: 20%
   Comments on program statements: 10%
   Answering oral questions: 10%
8. The deadline for submission: 09 October 2023, 12 PM.

PROBLEM 1: Music Player.

Implement a music player using a doubly linked list. The music player should be able to:

- ✓ Add songs to the playlist
- ✓ Remove songs from the playlist
- ✓ Play the next song in the playlist
- ✓ Play the previous song in the playlist
- ✓ Shuffle the playlist

Requirements:

Use a doubly linked list to store the songs in the playlist. The nodes in the doubly linked list should have the following fields:

Data: The song object

(The Song class can contain info about a song like title, artist, duration, album etc. In such a case the play() method will simply display that info along with a message that a song is being played rather than playing a real song.)

Next: A pointer to the next node in the list

Prev: A pointer to the previous node in the list

Implementation:

Implement the following methods:

- ✓ addSong(): Adds a song to the playlist.
- ✓ removeSong(): Removes a song from the playlist.
- ✓ playNext(): Plays the next song in the playlist.
- ✓ playPrevious(): Plays the previous song in the playlist.
- ✓ shuffle(): Shuffles the playlist.

The playNext() and playPrevious() methods should use the doubly linked list to traverse the playlist in the forward and backward directions, respectively.

The shuffle() method should randomly reorder the nodes in the doubly linked list.

Use the addSong(), removeSong() methods to manage the playlist.

To start playing the music, you can call the play() method.

The play() method will start playing the song at the head of the playlist. If you want to play the next song, you can call the playNext() method. To play the previous song, you can call the playPrevious() method.

You can also shuffle the playlist using the shuffle() method.

(Optional) Creative Extension: Will not add to grade. This is only for additional practice.

Students who can program well can consider adding the following to the application they built.

- ✓ Play a real Song rather than simply displaying a message about a song.
- ✓ Add a feature to the music player that allows the user to create and save multiple playlists.
- ✓ Implement a search feature that allows the user to search for songs in the playlist by title or artist.
- ✓ Add support for different audio formats, such as MP3, WAV, and FLAC.

PROBLEM 2: Cache Storage

Implement a doubly linked list-based cache. The cache should be able to store a fixed number of items. When a new item is added to the cache, the least recently used item should be evicted from the cache.

Requirements:

Use a doubly linked list to store the items in the cache.

The doubly linked list should have the following fields:

- ✓ Data: The item stored in the cache
- ✓ Next: A pointer to the next node in the list
- ✓ Prev: A pointer to the previous node in the list

Implementation:

Implement the following methods:

- ✓ AddItem(): Adds an item to the cache.
- ✓ GetItem(): Gets an item from the cache.
- ✓ EvictLeastRecentlyUsedItem(): Evicts the least recently used item from the cache.

The AddItem() method should add the new item to the head of the doubly linked list.

The GetItem() method should search for the item in the doubly linked list. If the item is found, it should be moved to the head of the list and returned. Otherwise, null should be returned.

The EvictLeastRecentlyUsedItem() method should remove the tail node from the doubly linked list and return the item stored in that node.