Introduction to Programming and Data Structures

Homework 2: Playlist Manager

OVERVIEW

In this assignment, you will implement a C++ program to manage a music playlist using a **singly linked list**. Your program will read an initial list of songs from an input file and then process a series of commands from a command file. This will test your skills in file I/O, advanced string parsing, and complex linked list manipulations like adding, removing, and reordering nodes.

INPUT FORMAT

1. Input File (input.txt)
The input file will contain an initial list of songs. Each song is on a new line and follows a strict format:
[title: value; artist: value; genre: value; duration: value]

- **title**, **artist**, and **genre** are strings that can contain spaces.
- **duration** is an integer representing seconds.
- The file can be empty.

2. Command File (command.txt)
The command file contains a sequence of operations to perform on the playlist.

- Add([song_details])
  Adds a new song to the end of the playlist.
  - Example: Add([title: Levitating; artist: Dua Lipa; genre: Pop; duration: 203])
- Remove(attribute: value)
  Removes all songs from the list that match the given attribute and value. The value for comparison will be an exact match.
  - Example: Remove(genre: Pop) would remove any songs with the genre "Pop".
- Group(attribute)
  This command reorders the list by grouping songs with the same attribute value together. The relative order of songs within a group must be preserved. To ensure a single correct output, the final order of the groups is determined by the first appearance of a song from that group in the list before grouping.
  - Example: If the list is Rock1, Pop1, Rock2, and you run Group(genre), the "Rock" group comes first because Rock1 appeared before Pop1. The final order would be Rock1, Rock2, Pop1.
- Print()
  This command has no arguments. When called, it should print the current state of the entire linked list to the output file.

EXAMPLE WALKTHROUGH

To clarify the logic, let's trace an example that shows the list changing.

input.txt

[title: Yesterday; artist: The Beatles; genre: Rock; duration: 125]
[title: Billie Jean; artist: Michael Jackson; genre: Pop; duration: 294]
[title: Hey Jude; artist: The Beatles; genre: Rock; duration: 431]

command.txt

Group(artist)
Print()

Step-by-Step Execution:

1. **Initial List (after reading input.txt):**
    1. [title: Yesterday; artist: The Beatles; genre: Rock; duration: 125]
    2. [title: Billie Jean; artist: Michael Jackson; genre: Pop; duration: 294]
    3. [title: Hey Jude; artist: The Beatles; genre: Rock; duration: 431]
2. **After Group(artist) command:**
    - The program looks at the list above. The first song is by **The Beatles**. This establishes that the "The Beatles" group will be **first** in the reordered list.
    - The second song is by **Michael Jackson**. This is the first time this artist appears, so the "Michael Jackson" group will be **second**.
    - The third song ("Hey Jude") is by "The Beatles". It must be moved to join the "The Beatles" group. Since it appeared after "Yesterday" in the original list, it will be placed after "Yesterday" in the final group to preserve relative order.
3. **Final List State for Print():** The list has been visibly reordered.
    1. [title: Yesterday; artist: The Beatles; genre: Rock; duration: 125]
    2. [title: Hey Jude; artist: The Beatles; genre: Rock; duration: 431]
    3. [title: Billie Jean; artist: Michael Jackson; genre: Pop; duration: 294]

OUTPUT FORMAT

Your program should generate an output file that contains the results of all Print commands. Each call to Print should output every song in the playlist at that moment, with each song on its own line.

RESOURCES & SUBMISSION

Please refer to the course website for detailed instructions on how to compile, test, and submit your homework on the server.

- **General Homework Information & Submission:**

https://uh.edu/nouhadrizk/about/courses/programming-and-data-structures/homework/

- **Testing Your Program:** https://uh.edu/nouhadrizk/about/courses/programming-and-data-structures/homework/homework-introduction/

You must submit your C++ source files (.cpp and .h) to a folder named **hw2** (case-sensitive) in your root directory on the server.

ACADEMIC INTEGRITY

This is an individual assignment. All work submitted must be your own. Your submission will be automatically checked for plagiarism. Any detected instances of copying or cheating will result in a grade of **0** and potential further disciplinary action.