CptS 223 – Advanced Data Structures in C++

Programming Assignment 1: Linux Commands Matching Game

Assigned: January 23, 2025

Due: February 2, 2025

Due with penalties: February 5, 2025

Weight: 6% of course grade

IMPORTANT NOTE: Your solutions must be your ***own*** work. Refer to the academic integrity statement in the syllabus.

Learning Objectives

By completing this assignment, you will:

- Develop a small C++ game (text-based trivia-style game).
- List and define 30 popular Linux commands.
- Construct a template-based linked list class.
- Work with .csv files in C++ (open, edit, parse, and save).
- Evaluate linked lists and arrays for data storage, discussing pros and cons in the context of this assignment.

Prerequisites

Before starting, ensure you can:

- Analyze requirements and design solutions using top-down principles.
- Design, implement, and test medium-scale programs in an object-oriented language.
- Use basic Linux commands to edit, compile, and execute programs.
- Implement and understand linked lists.

Assignment Overview

Create a text-based, single-player game where players match Linux commands with their descriptions. The game awards or deducts points based on correct or incorrect matches. Player profiles and scores must persist between sessions. So, the player has the option to save and resume the game at a later time.

Main Menu

Provide the following options to the user when the program starts/executes:

1. Game Rules: Display a brief explanation of the game rules.
2. Play New Game: Prompt for player name and number of questions. Present commands and descriptions as described above.
3. Load Previous Game: Load an existing player profile by name and display stored points.

4. Add Command: Allow new commands to be added to the linked list while avoiding duplicates.
    a. IMPORTANT NOTE: You should store the commands in a **Linked list container**. Instead of using the out-of-box containers C++ provides you, you need to implement a LinkedList container on your own. Also, this container must be **reusable** for a different problem, so instead of hardcoding what goes into the list, you need to use **templates**.
5. Remove Command: Prompt to remove a command from the linked list.
6. Display All Commands: Show all commands, descriptions, and points.
7. Save and Exit: Save commands and profiles, deallocate memory, and exit.

Technical Requirements

1. Linked List Implementation: Use a singly linked list with a class template for 'Node' and 'List' (*This should be your very own implementation*)
2. Avoid using C++ containers, available out of the box, like vector or list. The objective of the assignment is to test whether you can implement a linked list by yourself using templates.
3. Store player profiles in an array.
    a. Make sure that you specify the advantages and disadvantages of Array vs LinkedList in the readme file.

Development Instructions

1. Files: Include at least one header file and two source files, one named main.
2. Building and Running: Compile using g++ and run using ./PA1. Add these commands to a readme file.
3. GitHub/Git: Not used for this assignment due to operational concerns. Submit all files directly via Canvas.

Submission Guidelines

Submit all files (source code, header files, commands.csv, profiles.csv, and README) via Canvas.

Grading

This assignment is worth 100 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 3 pts (1.5 pts/advantage and disadvantage) for listing 1 advantage and 1 disadvantage of using a linked list for the data structure involved with storing the commands and descriptions; you must relate your ideas to the way the list is used in THIS assignment. This should be in the README file.
- 3 pts (1.5 pts/advantage and disadvantage) for listing 1 advantage and 1 disadvantage of using an array for the data structure involved with storing the user profiles; you must relate your ideas to the way the list is used in THIS assignment. This should be in the README file.
- 25 pts for developing a correct class List, template with 1 parameter
    - 6 pts for correct insertAtFront()– needed when loading from commands.csv

- 7 pts for correct removeNode()– needed when removing commands from the list
- 8 pts for other functions needed to operate on the list
- 2 pts for correct data members
- 2 pts for correct constructors/destructors
- 7 pts for correct implementation of class Node template with 1 parameter
- 5 pts for correct implementation of concrete class Data

- 3 pts for satisfying the *Main Menu Requirement*
- 18 pts for satisfying the *Play Game Requirement*
  - 2 pts for correctly prompting user for the number of questions.
  - 7 pts for correctly generating a random command and 3 random descriptions
  - 3 pts for verifying answer
  - 2 pts for updating player points
  - 4 pts for generating correct number of questions
  - 5 pts for satisfying the *Load Previous Game Requirement*
  - 5 pts for satisfying the *Add Command Requirement*
  - 5 pts for satisfying *Remove Command Requirement*
  - 5 pts for satisfying *Display All Commands Requirement*
- 10 pts for satisfying the *Exit Requirement*
  - 4 pts for correctly writing to commands.csv
  - 3 pts for correctly writing to profiles.csv
  - 2 pts for deallocating linked list memory
  - 1 pt for closing the files
- 3 pts for the appropriate class and top-down design
- 3 pts for adherence to **proper programming style** established for the class and comments

Notes

- Explore Linux commands at GeeksforGeeks.
- Focus on base commands without options (e.g., `ls` not `ls -l`).
- Start with the following commands: `pwd`, `ls`, `cd`, `mkdir`, `rmdir`, `rm`, `cp`, `mv`, `ssh`, `scp`, `man`, `g++`, `gcc`, `make`, `ps`, `kill`, `top`, `who`, `chmod`, `cat`, `alias`, `chown`, `df`, `grep`, `echo`, `find`, `clear`, `diff`, `env`, `free`.

Tips

Test thoroughly and check edge cases. Use meaningful comments and adhere to class coding standards.