COSC2436 hw1: Sorting Algorithms with Linked List

1. Introduction

Given a list of credentials as input, you will need to implement a C++ program to add these input objects into a linked list. Within the linked list, your program needs to perform different adding, removing and sorting operations base on the given commands. This homework will focus on linked list implementation and simple sorting techniques. When submit your assignment, please name the folder on the server as "hw1". In this HW we are making a simple data base for a casino!

2. Input files

- The input file will contain a list of credentials (ranging from 0 to 100).
- This file can be empty.
- Each credential represents a node in the linked list and should be added one by one to the end of the linked list.
- Each credential will have four attributes: Name, age, deposit and number of chips. Note: Name will always contain alphabet character (a – z, A- Z), no spaces or special character included. Age is an integer between 21 and 80. Deposit is an integer between 100 to 10000. Number of drinks is between 0 to 10.

  The formatting of each credential is as follow:

  [name: value; age: value; deposit: value; number of drinks: value]

- You can safely consider that all the inputs are in the correct format.
- In the case when the input is empty, continue to process the command.
- While reading the input, \n and \r should be removed before processing string.
- Input might contain duplicate id credential or duplicate username credential, please read section 5 below on how to process duplicate cases.

3. Command files

- There will be three types of command: Add, Remove, and Sort. The commands should be executed in the order that they appear (latter command should always run based on the former command's result).

  o Add (index) [credential]

  - ✦ The Add command will be followed by an integer inside parenthesis (represent the index in the linked list to be added) and then followed by a credential.
    For instance: Add (2) [name: Rob; age: 34; deposit:1234; number of drinks: 3]
    Will add Rob after the second node of the list.
  - ✦ If index = 0, meaning the credential should be added at the beginning of the linked list.
  - ✦ If the index >= size of the linked list, meaning the credential should be added at the end of the linked list.

- Add command might contain duplicate id credential or duplicate username credential, please read section 5 below on how to process duplicate cases.

o Remove (attribute: value)

- The Remove command will be followed by an attribute and its value (placed inside parenthesis).
- Every credential that contains the attribute with the matching value should be removed from the linked list.

- Ex: Remove [number of drinks: 4]
  remove every credential that contain value "4" in attribute "number of drinks".

o Sort (attribute)

- Sort the linked list base on the given attribute (placed inside parenthesis).
- Sort ascending for deposit. Sort (deposit)
- Sort alphabetically for name
- When sorting if two values are similar, don't swap their position.

- The command can be empty.
- While reading the command, \n and \r should be removed before processing string.

4. Output files

- The output file should display every credential in your linked list after executing all the command in the command file.
- Each credential will be on its own line.

5. Adding Operations

- When adding a credential to the linked list, the credential might contain duplicate id or duplicate username.
- In the case when the credential contains duplicate name and age, update the previous credential's attributes with the new credential's attributes.

o Ex: If the credential [name: Ali; age: 26; deposit: 7500; number of drinks: 3] is in the linked list when adding [name: Ali; age: 26; deposit: 3000; number of drinks: 5],

since the two credential have matching name and age you will update the deposit and number of drinks of the credential in the linked list to the latter credential.

6. Requirements

Homework is individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. Code that is copied from another student (for instance,

renaming variables, changing for and while loops, changing indentation, etc. will be treated as copy) will be detected and result in ”0” in this homework. The limit is 50% similarity.

7. Turn in your homework

Homework 1 needs to be turned in to our Linux server, follow the link here https://rizk.netlify.app/courses/cosc2430/2_resources/

Make sure to create a folder under your root directory, name it “hw1” (case sensitive), copy all your .cpp and .h file to this folder, “ArgumentManager.h” need to be included as well.

PS: This document may have typos, if you think something illogical, please email TAs for confirmation.