COSC2430 Homework 2: Linked List

1. Introduction

You will create a C++ program to manage problems on Leetcode (a site to practice coding interviews). You need to store data from input files and perform required actions from the command files including adding, removing, and sorting.

The purpose of this homework is to get students familiar with Linked List and simple sorting techniques. Please name the folder on the server as “hw2”.

2. Input and Output

a. Input file

The input file contains a list of problems in the similar format:

i. problem_id:*value*, problem_name:*value*, difficulty:*value*
ii. Difficulty will always be either Easy, Medium, or Hard.
iii. Files may contain empty lines.

b. Command file

The command files contains a list of commands:

i. add pos:*value* book_id:*value*, book_name:*value*, book_author:*value*
ii. remove keyword:value
    - Keyword is one of: *pos, problem_id, problem_name,* or *difficulty*
iii. sort keyword direction
    - Keyword is one of: *problem_id, problem_name,* or *difficulty*
    - Direction is one of: *increasing* or *decreasing*
iv. Files may contain empty lines.

c. Output file

The output is a single text file. You will need to output the list of nodes in the Linked List line by line in the sequential order.
problem_id:*value*, problem_name:*value*, difficulty:*value*

d. Examples

i. Example 1

input21.txt

problem_id:321038, problem_name:Binary Search Tree to Greater Sum Tree, difficulty:Medium
problem_id:580101, problem_name:Shortest Unsorted Continuous Subarray, difficulty:Easy
problem_id:297978, problem_name:Find All Numbers Disappeared in an Array, difficulty:Easy
problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard
problem_id:297985, problem_name:Validate Binary Search Tree, difficulty:Medium

command21.txt

sort problem_id increasing
add pos:2 problem_id:440445, problem_name:Unknown, difficulty:Hard
add pos:2 problem_id:321038, problem_name:Binary Search Tree to Greater Sum Tree, difficulty:Medium

output21.txt

problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard
problem_id:297978, problem_name:Find All Numbers Disappeared in an Array, difficulty:Easy
problem_id:440445, problem_name:Unknown, difficulty:Hard
problem_id:297985, problem_name:Validate Binary Search Tree, difficulty:Medium
problem_id:321038, problem_name:Binary Search Tree to Greater Sum Tree, difficulty:Medium
problem_id:580101, problem_name:Shortest Unsorted Continuous Subarray, difficulty:Easy

Linux Command:

./list “input=input21.txt;command=command21.txt;output=output21.txt”
./list input=input21.txt command=command21.txt output=output21.txt

ii. Example 2

input22.txt

problem_id:580101, problem_name:Shortest Unsorted Continuous Subarray, difficulty:Easy

problem_id:297978, problem_name:Find All Numbers Disappeared in an Array, difficulty:Easy
problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard

problem_id:440446, problem_name:Median of Two Sorted Arrays, difficulty:Hard

command22.txt

sort difficulty decreasing
add pos:2 problem_id:297985, problem_name:Unknown, difficulty:Medium
remove difficulty:Medium
add pos:4 problem_id:267101, problem_name:Remove Duplicates from Sorted List II, difficulty:Medium

output22.txt

problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard
problem_id:440446, problem_name:Median of Two Sorted Arrays, difficulty:Hard
problem_id:580101, problem_name:Shortest Unsorted Continuous Subarray, difficulty:Easy
problem_id:297978, problem_name:Find All Numbers Disappeared in an Array, difficulty:Easy
problem_id:267101, problem_name:Remove Duplicates from Sorted List II, difficulty:Medium

Linux Command:

./list “input=input22.txt;command=command22.txt;output=output22.txt”
./list input=input22.txt command=command22.txt output=output22.txt

iii. **Example 3**

input23.txt

problem_id:321038, problem_name:Binary Search Tree to Greater Sum Tree, difficulty:Medium
problem_id:580101, problem_name:Shortest Unsorted Continuous Subarray, difficulty:Easy

problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard
problem_id:355828, problem_name:Count Unique Characters of All Substrings of a Given String, difficulty:Hard

problem_id:297985, problem_name:Validate Binary Search Tree, difficulty:Medium
problem_id:440446, problem_name:Validate Binary Search Tree, difficulty:Medium

command23.txt

remove problem_id:440446
remove problem_name:Validate
remove difficulty:Easy
remove pos:1
remove difficulty:Hard
remove pos:9
remove pos:0
remove problem_id:297985
add pos:2 problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard
add pos:3 problem_id:123297, problem_name:Queue Reconstruction by Height, difficulty:Medium

output23.txt

problem_id:123297, problem_name:Serialize and Deserialize Binary Tree, difficulty:Hard

Linux Command:

./list “input=input23.txt;command=command23.txt;output=output23.txt”
./list input=input23.txt command=command23.txt output=output23.txt

3. The Rules and Operations

- You will need to read the input file containing a list of problems. Each line will represent a problem and its data (id, name, difficulty). You need to create a Linked List and add each problem sequentially.

- In the command file, there are multiple commands, and each line represents a single command. You need to read commands and perform it in the sequential order one by one.

- For the “add” command:
  - Index starts from 0.
  - If you get an index that is negative, you will need to add it in the beginning of the list.
  - If you get an index that is greater than list size, you will need to add it at the end of the list.
  - If you get a problem_id that is already existing in the list, you will **NOT** add the problem.

- For the “remove” command:
  - Index starts from 0.
  - Remove by difficulty: remove all problems having the given difficulty
  - If you get an invalid index (out of bound), you will **NOT** need to perform any action.

- For the “sort” command:
  - Sort by difficulty: if two problems have the same difficulty, keep the same order as the input files.
  - Sort by problem_name: compare the strings
  - Sort by problem_id: compare the ids

- You may need to handle a maximum of 1000 problems and 100 commands.
- If you read an empty line, you will ignore it.
- Sorting will be either **decreasing or increasing** order. You will need to sort it as required in the commands.
- All problems are **case sensitive**.

4. Requirements

Do **NOT** use the list in STL. Please create the Linked List manually.
The main C++ problem will become the executable to be tested by the TAs. The result file should be written to another text file (output file), provided with the command line.

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. Here are some previous homework which have been found to copy each other (the main function has been deleted).

5. **Turn in your homework**

   Homework 2 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 hw2 (name need to be lower case), only copy your code to this folder, no testcase or other files needed.

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