COSC2430 Hw1: Name list management (Linked lists practice)

1. Introduction

You will create a C++ program to manage a name list for class selection. Read a file that has a lot of items that stand for the information of a name list of certain class. Then sort the records by specific attributes in alphabetical order, and output to a plain text file.

2. Input and Output

The record file is very clean (but may contain duplicated records) and have uniform attributes(columns). Each record is in one line (ends in '\n') without any spaces. Each record is within the brace "{" and "}" characters, each attribute in one record is separated by comma "," character, each attribute consists of key and value (separated by colon ":" character). When adding an item, just use the information of that record. When deleting an item, use "delete id".

The command file is a short one, only commands like <id | first | last | DOB | GPA> are included, notice that each line only contain one command, every result should base on the sequence of commands, the command should be executed from top to bottom.

When output a file, the format is the same as input file, one line has only one record, each line ends with '\n'.

Notice that duplicate record should only be add once. Deleted item should not be shown in output unless it is added again. All records should be processed sequentially from begin to end.

3. Program and input specification

The main C++ program will become the executable to be tested by the TAs. The result file should be write to another text file (output file), provided with the command line. Notice the input and output files are specified in the command line, not inside the C++ code. Notice also the quotes in the program call, to avoid Unix/Windows get confused.

The general call to the executable is as follows:

sort "input=input1.txt;output=output1.txt;sort=sort1.txt"

Call example with another command line type.

sort input=input1.txt output=output1.txt sort=sort1.txt

The items in the smaller than and greater than characters "<" and ">" can only choose one value.

Example 1 of input and output

input11.txt
{id:1234567,first:Mary,last:Green,DOB:1996-10-03,GPA:4.0}
{id:1234568,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
{id:1234587,first:Katy,last:Green,DOB:1995-08-18,GPA:4.0}

sort11.txt
id

Command line:
```
sort input=input11.txt output=output11.txt sort=sort11.txt
```

output11.txt
```
{id:1234567,first:Mary,last:Green,DOB:1996-10-03,GPA:4.0}
{id:1234568,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
{id:1234587,first:Katy,last:Green,DOB:1995-08-18,GPA:4.0}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
```

Example 2 of input and output

input12.txt
```
{id:1234567,first:Mary,last:Green,DOB:1996-10-03,GPA:4.0}
{id:1234568,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
{id:1234587,first:Katy,last:Green,DOB:1995-08-18,GPA:4.0}
delete 1234568
{id:1234570,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
```

sort12.txt
```
id
DOB
```

Command line:
```
sort input=input12.txt output=output12.txt sort=sort12.txt
```

output12.txt
```
{id:1234587,first:Katy,last:Green,DOB:1995-08-18,GPA:4.0}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
{id:1234567,first:Mary,last:Green,DOB:1996-10-03,GPA:4.0}
{id:1234570,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
```

Example 3 of input and output

input13.txt
```
{id:1234567,first:Mary,last:Green,DOB:1996-10-03,GPA:4.0}
{id:1234568,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
{id:1234587,first:Katy,last:Green,DOB:1995-08-18,GPA:4.0}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
```

```
sort13.txt
id
```

Command line:

```
sort input=input13.txt output=output13.txt sort=sort13.txt
```

```
output13.txt
{id:1234567,first:Mary,last:Green,DOB:1996-10-03,GPA:4.0}
{id:1234568,first:Peter,last:White,DOB:1997-05-22,GPA:3.8}
{id:1234587,first:Katy,last:Green,DOB:1995-08-18,GPA:4.0}
{id:1654238,first:Nick,last:Park,DOB:1995-08-18,GPA:4.0}
```

4. Requirements

- C++:

  It is encouraged you do not use existing STL, vector classes since you will have to develop your own C++ classes and functions in future homework.

  Your C++ code must be clear, indented and commented.

  You must create a struct or a class to save each whole record and the elements of the record. The whole record is used for eliminating duplicate ones while the elements of the record are used for sorting.

  There are 200 records at most, so if you want to create a fixed memory space, that is the number. Every time when deleting a record, you should adjust the pointer to its next, so that you won't lost all record followed.

  Include comments on each source code file and a README file on how to compile/run with examples, as obvious it may seem.

  Your program will be tested with GNU C++. Therefore, you are encouraged to work only on Unix. You can use other C++ compilers, but the TAs cannot provide support or test your programs with other compilers.

- Output:

  The output file must contain the result, in the same format as input. Pay attention to the order of elements, disordered elements would be treated as wrong answer.

  Your program should write error messages to the standard output (cout, ptintf).

5. Hand over your homework

Homework 1 need to be handed over to our Linux server, follow the link here http://www2.cs.uh.edu/~rizk/homework.html.

Make sure to create a folder under your root directory, name it hw1 (name need to be lower case), only copy your code to this folder, no testcase or other files needed.

ps. This document may has typos, if you think something illogical, please email TAs or Teacher for comfirmation.