COSC2430 Hw2: Name list management (Linked lists practice)

1. Introduction

You will create a C++ program to manage a name list. 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 to alphabetical ascending order, and output to a plain text file.

2. Input and Output

a. Input file

1) The input file has multiple records (number > 1). Your program should read the records one by one from the beginning of the file to the end.
2) Each record has uniform attributes (columns), the attributes appear in a fixed order, a record should contain all the keys, no empty values part will be given. But the input file may contain duplicated records. If two records have same id value, they can be recognized as duplicated records. You should always update record with the latter one. If somebody's name has more than one words, the separator would be underline "_". Other than useful information in records, no other character or space will be given.
3) Each record takes one line (ends in '\n'), and one line only contains one

record. Each record is enclosed by the brace “{“ and “}” characters. Attributes in records are separated by comma “,” character, each attribute consists of key and value (separated by colon “:” character).

4) The id value always uses 7 digits. DOB values always with the format YYYY-MM-DD, which are reasonable date. GPA value always with the format X.Y, X is the unit number, Y is the decimal number.

5) All the keys are case sensitive and with the exact words shown below in example. They will not change between different test cases. All the values should be outputted exactly as input, you should not change them to upper or lower case or truncate them.

6) When adding a record, use the information of that record. When deleting a record, use “delete id”. For one record, the id is the only key word, which cannot appear twice in your output file. Notice that duplicate record should only be added once. Deleted record should not be shown in output file unless it is added again. All records should be processed sequentially from begin to end.

7) The maximum number of records is 2000, if you want to use array to store it, that’s the limit.

b. sort file

1) The sort file only contains commands: id, first, last, DOB and GPA (case sensitive), there will be no space in sort file. Each line ends in ‘\n’. Notice that each line only contains one command, and one command only

contained in one line.

2) The outputted result should base on the sequence of commands, the commands should be executed from top to bottom sequentially. The latter commands should always base on the former commands' output.

c. output file

1) The output file should be clean, one record in one line, the line should end in '\n'. No space in records.

2) The outputted records in the output file should keep the same keys and values of input file, the only differences are the outputted file is ordered.

3) If two or more records by passing all commands have same order, the one with lower id value should be outputted first.

3. Program specification and Examples

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. All the necessary parameters will be in the command line when calling your program, so you don't need to worry about the missing file name problem. When calling your program, the format would be one of the two standard types as below, no mixed calling type will be given. 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

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:1995-08-28,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:1995-08-28,GPA:4.0}

4. Requirements summary

- 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 2000 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 in each source code file.

Your program will be tested with GNU C++. Therefore, you are encouraged to work 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 attributes, disordered attributes would be treated as wrong answer.

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

  Your program should output result within 5 seconds, after will be killed by system.

5. Turn in your homework

Homework 2 need to be turned in 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 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 or Teacher for confirmation.