CMSC 3613

Programming Assignment: Lists

Due Date: Check the D2L calendar for the due date.

Assignment:

Implement a program that uses a list (`list.h`) to manage a sequence of personal records. The format of the personal records is defined as a `struct` in `main.h` as follows.

```
struct Personal_record {
   string first_name;
   string last_name;
   string ID;
};
```

Please use the main program provided by the instructor. You can use either of the list implementations (either linked implementation or contiguous implementation by removing or commenting one of the include statement on the very top of the `main.cpp` file). The following functionalities should be implemented in the program as marked by "write code to implement Requirements *" in the comments of `main.cpp`.

1. Insert every record read from a data file into the list (`record_list`) declared in `main.cpp` in **ascending alphabetical order** based on the last name of the record. For records with the same last name, break the tie using the first name. If a record already exists with the same ID, or with the same pair of first and last names, the new record should be discarded instead of being inserted into the list.
2. Print the content of the list using the following format for each record through the provided function `void traverse(void (*visit) (List_entry &))` in `List.h`. The format of each record should be:
   ```
   Last Name: "real-last-name-of-the-record"
   First Name: "real-first-name-of-the-record"
   ID: "real-ID-of-the-record"
   ```
3. Support search functionality for a record in the list based on the ID or the pair of the first and last names. Display the record using the same format in Item 2 above. If the record is not found, display "`Not found`".

Bonus tasks:

1. In the Sorting task in Item 1 and Search task in Item 3, we need to compare two records, and we need to consider the comparisons of the first and last names. A naïve solution is to compare the last name and then compare the first name in nested if-else statements. Can you provide a better solution without nested if-else statements (separating the comparison of last name and first name into two functions is still counted as nested if-else statement)?
2. Support the basic fuzzy search functionality. If the search is based on the pair of the first and last name, and the searched name is the prefix of a record in the list, then we will also print this record, e.g., the user inputs "John S" (John is the first name and S is the last name) and "Johnson Smith" is a record in the list, then we should also print this matched record "Johnson Smith". Please pay attention: multiple records can be matched in the output.

Requirements:

1. Fill in the code segments required in the `main.cpp` file: 3 segments corresponding to 3 tasks above.
2. The program is menu-driven. You can import a list of records from the given data file.

Test Files:

One test data file "data.txt" is provided.

Submission:

1. Please provide a readme file, to help with the compilation and execution of your code. For example, information about your operating system and detailed command to compile your code should be included.

2. Submit a report, which should include the following items:
   1) Your name and UCO email address
   2) The project number, i.e., p01
   3) Include your project progress, by referencing the ProjectPlan.docx file.
   4) A brief discussion
      - Which version of the list implementations is better: contiguous or linked? Please explain.
      - a) How to understand the function: `void traverse(void (*visit) (List_entry &))`?
      - b) What's the search algorithm you used in Item 3? Briefly explain why it's efficient.
      - c) Can you directly use the comparison operators (e.g., >, <, >=, etc.) to compare instances of Personal_record? If not directly supported, can you provide a solution to make it possible? Briefly explain your design.
   5) A screenshot of a test run.

3. Your source code also needs to submitted (e.g., .h, .cpp, makefile, etc).

4. All the files need to be zipped as **p01_group*.zip**, where * means your group number, e.g., the submission from group 3 should be named as p01_group3.zip.

Evaluation:

This project will be evaluated according to the correctness of the various tasks specified above, and secondarily according to the quality of your code. The rubric is as follows:

Categories	**Weights**
`Task 1`	35%
`Task 2`	15%
`Task 3`	30%
`Report`	20%: 5% for each of 3 questions in report item 4), 5% for the rest.
`Bonus`	10% * 2

Notes:

1. To be considered on time, the program must be turned in before the due date.
2. Programs must reflect your knowledge and work, not others'.
3. No points, zero (0), will be given to any program containing compilation errors.