New York University Abu Dhabi
CS-UH 1050–Spring 2019
Programming Assignment 3
Out: Friday, April 19, 2019
Due: Thursday, May 09, 2019 at 17:00

Preamble:
In this project, you will develop a hotel-finder application termed `hotelFinder` that can help you search for a hotel in a specific city. The identification of hotels will be based on the key produced by the *combination* of `hotelName` and `cityName`.

Information of hotels is to be inserted/read from either the `tty` of `hotelFinder` or from a text file. The application should support deletion of hotel records as well. You will have to use the `Linux/FreeBSD C++/C` development environment.

Overall Description:
`hotelFinder` should manage (i.e., store, retrieve, and delete) records of hotels. Each such record is provided in ASCII format and spans at most one line of text. The line in question consists of the hotelName, cityName, stars, price, countryName and address. Examples of such lines depicting records of each hotels are:

```
hotelName,cityName,stars,price,countryName,address
Seabreeze Resort,Upolu,3.5,256,Samoa,Paradise Cove Aufaga
Princess Tui Inn Apia,Apia,2,99999,Samoa,1 Vaiala Beach Road PO Box 9590
Hosteria Las Lengas - Tierra del Fuego,Tierra del Fuego,2.5,99999,Chile,Lago Blanco
Hosteria Les Eclaireurs,Ushuaia,3,49,Argentina,Staiyakin 2676
..............
```

Through its prompt, the `hotelFinder` is expected to accept a number of commands outlined below; these commands accept as operand a string that provides a hotel's data to be inserted, or the combination of `hotelName` and `cityName` used as the key to look up records. You may assume that the input to `hotelFinder` is always syntactically correct. The `hotelFinder` commands are the following:

- `find k`: Find the element having a key `k`[1] and display the entire record. Also, print out the number of comparisons made and the actual *time* taken by the find execution.
- `add s`: The data provided by string `s` consisting `hotelName, cityName, stars, price, countryName, address` corresponding to a hotel are to be inserted. If the element already exists, do not add it again, but simply print out a warning to the standard error.
- `delete k`: Delete the hotel record with key `k`[1]. If no such element exists, print out a warning message to the standard error.
- `dump f`: Dump the content of the entire structure(s) into the file `f` and sort the content of the file `f` according to the `hotelName` in alphabetically increasing order.
- `allinCity c`: List all hotels in the city `c`.
- `quit`: terminate the program with graceful release of all dynamically acquired memory.

[1] *combination of* `hotelName` *and* `cityName` separated by a comma

Requirements:
While designing your program, you will have to address the efficiency of all the supported operations. You should explain in your write-up the design choices made and how they relate to efficiency. In particular you *should estimate the asymptotic running time* that you expect from your structure(s) per operation. Your overall objective should be to create a modular, easy–to–understand, efficient, and well-commented program.

Efficiency means among other things that your solution should scale well. The number of distinct cities in an input file could be in the range of thousands. If your code is passably fast for an input with a few hundred hotels which are located in 5 different cities this does not mean that you will get full credit. Marks will be deducted for inefficient implementations of queries. Efficiency dictates that you *must* use hashing as your underlying structures. You should implement yourselves the data structure instead of using the version in the `C++` STL library. This will count towards 30% of your grade. Ensure first that your choice promotes efficiency.

Procedural Matters:
◇ Your program is to be written in `C++` and must run on `a Unix/FreeBSD` operating system.
◇ Khalid Mengal (`khalid.mengal-AT-nyu.edu`) will be responsible for answering questions as well as reviewing and marking the assignment.

How to Invoke your Application:
The application should be invoked as follows:

```
mymachine-prompt >>  ./hotelFinder -f <filename>
```

where the file `<filename>` contains records to be inserted into your data structure(s).

Group submission:
You should submit your work in groups of two. Both members of a group should be able to answer questions about *any* aspect of the project. To form a group, please send an email to Khalid Mengal **before** submission with the names and NetIDs of the group members. **Both** members of the group should each submit the same set of files by the deadline.

What you Need to Submit:

1. A directory that contains all your work including source, header, `Makefile`, a `README` file, etc.
2. A short write-up about the design choices you have taken in order to design your program(s); two (at most three) pages in PDF format are expected.
3. All the above should be submitted in the form of a `zip` or `tar` file bearing your name (for instance `AndreBreton-Proj3.tar`).
4. Submit the above zip/tarball using *NYUclasses*.

Grading Scheme:

Aspect of Programming Assignment Marked	Percentage of Grade (0–100)
Quality of Code Organization & Modularity	10%
Hashing implementation	30%
Correct and **efficient** execution of Queries	50%
Use of Makefile & Separate Compilation	5%
Well Commented Code	5%

Noteworthy Points:

1. You have to use *separate compilation* in the development of your program. In other words your code should be distributed in at least two `cpp` files (potentially more).

2. We will provide some sample input files on *NYUclasses*. Source: `https://github.com/lucasmonteiro001/free-world-hotel-database`

3. Although it is understood that you may exchange ideas on how to make things work and seek advice from fellow students, sharing of code is *not allowed*.

4. If you use code that is not your own, you will have to provide *appropriate citation* (i.e., explicitly state where you found the code). Regardless, you have to fully understand what and how such pieces of code do and explain why you resorted to using them. There are tools for detecting network-wide code similarity and we do use them.