CS 284: Final Project

Due: December 14, 11:55pm

Collaboration Policy: Projects are to be done individually: each student must hand in his or her own work. It is acceptable for students to collaborate in understanding the material but **not** in programming.

Use of the Internet is allowed, but that does not include searching for existing solutions. **Under absolutely no circumstances** code can be exchanged among the students.

If some code was shown in class, it can be used, but it must have been obtained from Canvas, the instructor, or the TA. In addition, you may use the code in the text book. **In all such cases you must clearly attribute the source, as required by Stevens Honor Code!** (For example, you may say that you are using the code on pp. x-y of the book, or the code from Slide x of Lecture y, or the code on Canvas at the URL *z* posted by TA on date t.)

Note well: For this specific assignment, it is **not** allowed to use the Java Collections.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

Synopsis

You work for the IT department of a university. In this capacity you receive messages from users reporting problems with the various servers that are available to them. Each message is of the form:

Username|*Hostname*|*Problem*|*Level.*

Based on the priority level assigned by the user and the user's own history (see below), a priority will be assigned to each message and it will inserted into a priority queue.

Messages with higher priority are to be processed first. Users can assign priority levels between 1 and 5 to their messages. Any message whose level is outside of this range should be rejected.

At the end of each batch of messages, a line containing only the word *break* in lowercase will mark the end of the current batch. (The word *break* may not be the username of any user of the system.)

Once a batch of messages have been inserted into the priority queue and a break has been reached, the messages should be processed according to their respective priority values.

After you remove a message from the priority queue, you should assign to it a unique ID number and insert it to the end of a long archive containing all messages that have been processed. (There is one exception under which a message is not inserted into the archive. It will be mentioned later.)

As the archive grows, it will become unmanageable, and so there is a need to retrieve messages from the same user (concerning the same host or the same problem type) quickly, that is in a time independent of the size of the archive. This will be useful for the tasks below.

When you remove a message from the priority queue, the program should act as follows:

- If a message from the same user reporting the same problem about the same host had been processed any time in the past (i.e., the message exists in the archive), it should be ignored.
- If the type of the problem in the message has been fixed on the same host in the past, an encouraging message should be printed informing the user that you know how to fix the problem.
- If the type of the problem in the message has been fixed on a different host in the past, a somewhat less encouraging message should be printed.

(Note that the priority level is irrelevant for all the above tests).
After that, the next batch of messages should be read from the input file and the above process should be repeated.

Requirements

Input

Your program should read from a file named "messages.txt" that contains the information provided by the users. Each line has the following format:

Username|Hostname|Problem|Level

(For example, a line may look like that:

alecsmart|arts|RAM|4

(To parse the above, you may re-use the string-parsing examples in the book.)

Your code should also check for the *break* line that should trigger the processing of messages in the priority queue.

Data Structures

You should implement a priority queue for inserting the messages once they have been read from the input file. The message with the highest priority should be returned first.
There should also be an array that serves as the archive of all messages that have not been ignored.

Finally, there should be hash tables, implemented using open addressing and linear probing, which enable fast retrieval of messages based on username, hostname or issue type. The hash table should store the ID number of each message only. Given the ID number, gathering all relevant data from the archive should be easy.

Message Priority

The internal priority of a message should be determined by the level set by the user and adjusted by the user's complaint history as follows:

$$Priority_{PQ} = Priority - 0.05 \times TotalLevels_{user},$$

where *TotalLevels*$_{user}$ is the sum of the levels of all messages filed by the user who issued the current message and can be found in the archive. For example, if a user has issued three messages whose level values were 2, 4 and 4, then the value of *TotalLevels*$_{user}$ is 10.

Hash Function

You should implement the following hash function (which is a variation on the built-in Java hash function for Strings):

$$h(name) = 23^{n-1} \times name[0] + 23^{n-2} \times name[1] + \cdots + name[n-1] = \sum_{i=0}^{n-1} 23^{n-1-i} name[i],$$

where n is the number of characters in the string.

1. This project is allocated a month for a reason, so start early!
2. Uppercase and lowercase characters should be treated as distinct.
3. Make sure your code can handle a name of an issue that contains spaces.
4. You can assume, however, that our input files will not be corrupt.
5. Use *long* instead of *int* variables for the hash codes.
6. Keep in mind that the hash table may require reallocation. You do not need to worry about
7. having the length of the new table be a prime number during reallocation, but ensure that the number is odd.

Submission requirements:

Please zip and submit the entire directory that contains your project and input files, excluding the bin directory. There will be a penalty for failing to do this.