Computer Science II — Homework 3 — Internet Ads.

Overview

This assignment is due Thursday, February 7 at 11:59:59pm and is worth 75 points toward your homework grade.

Internet search engines earn money through "sponsored ads" that appear in response to queries. Ads are generated by matching search terms from a query against search terms chosen by a sponsor. The engines keep track of how often ads are posted and how many of them led to user "clicks", charging sponsors differing amounts for posts and clicks.

Making this work efficiently requires some fairly sophisticated data structures. For this assignment, however, we are only going to use the simple data structures — vectors of class objects — we have learned about so far. Techniques we learn about later in the semester can be used to make these methods much more efficient.

Input

Once again, we will simulate the input to our system through a sequence of operations from an input file, and outputs will be to a second file. As in HW 2, your program should echo all input to an output file, preceeded by a blank line.

- **add keyword user url post-cost click-cost**
  Add the specified keyword for the user and indicate the url to be posted. The keyword, the user and the url will each be a single string. The post-cost and the click-cost will specify the charges for the user for each post and for each click. Several special cases must be handled:
  - If the user does not exist in the system, the user must be added. Output a simple message indicating that the user was added, then, on a separate line indicate that the keyword and url were added for the user.
  - If the user exists, but the keyword is new for this user, then the keyword and its associated post cost and click costs are added for this user/keyword combination. Output a message indicating that the keyword and url were added for the user.
  - If the user exists and already has this keyword, then the url, post cost, and the click costs must be changed for this user/keyword combination. Any total costs for this user and this keyword remain unchanged. Output a message indicating that the url was changed for the user and keyword combination.

- **search keyword**
  In response to a search for the given keyword, all users requesting an ad for this keyword must be found, and these ads must be "posted", which means the url must be output to the output file. When more than one ad is to be posted for this keyword, the order of the output must be the order of the "post-cost", with the ad having the highest post-cost coming first. If no ads are generated, output `<none>`.

- **click keyword url**
  Based on the given keyword, the url has been "clicked". Therefore, your program needs to find the user with the given keyword and url, and record that a click has occurred. The output in response this data should indicate the user charged.

- **remove keyword user**
  For the given user and keyword, mark the keyword inactive so that no ads are generated for this keyword and user. Do not remove the information about this keyword/user. In particular, this could be re-activated through the **add** command above. The output should indicate that the remove operation was successful.

- **statistics user**
  Generate statistics on the posted ads, the clicks and the total cost for the given user. Following the echoing of the input line, there should be one output line for each keyword and a final output line giving the total amount of money owed by the user. The line for each keyword should output the keyword, the (current) url, a 0 or 1 depending on if it is active, the number of ads posted, the number of clicks, and the cost. Be careful to observe that the cost per ad and per click can change in the middle of the program, so you need to keep track of the cost for each keyword as ads are generated and clicks are counted rather than waiting to the end. The order of the output should (simply) be the order that the keywords were first input to the system for the user. If the user does not exist, simply output `<none>`

All input will be correctly formatted. As always, please follow the output format of the examples posted on the course website as carefully as possible.

Classes

The first challenge here is the overall organization of the data. There are several possibilities. One is to have a vector of keywords **and** a separate vector of users, with some information repeated between them. For each keyword there would be a sequence (vector) of ads, with each ad having the appropriate information, including the information about the users. The actual costs would be stored with the users. A second choice and the one we recommend is to have a vector of users, where each user has a vector of ads. This is easier to think about and organize, but it is not particularly efficient.

If you follow this second route, you still have a significant challenge, because you will need two important classes. One must represent the information for an ad, including the keyword, the url, the cost information, whether or not it is active, and information on clicks and costs. The second will represent the user, including a vector of ads. If you get these implemented well, the rest of the project should be straightforward. The main program and its supporting functions will need a vector of users, and you will also need a simple class to store the ad information temporarily before posting it. You may define this latter class in the main program, but the other two (and larger) classes should each be in their own .h and .cpp files.

In order to help you, we have provided the solution `main.cpp` file. You may use as much or as little of this in your own solution, without any worry about academic integrity issues.

Submission

You will likely be submitting five C++ files — two .h files and three .cpp files — plus a `readme.txt` file. (See the homework assignment page for a template on the latter.) Even if you use our `main.cpp` without any changes, you must submit it. Please zip all these files together prior to submission, following the instructions on the course web page.