**Algorithms – Fall 2013**

**Reference Web Assignment**

**CS 4050**

Note: all updates appear together, at the end of the document. The original specs have not been changed.

Using a graph or other data structure that simulates a graph (hashes, maps, etc.), create a web of relationships that records letters of reference sent between various people.

Operations.

* read web from file (& possibly add to currently-open web)
* write web to file in input format (see below)
* output: all except the first two can go to the screen or a file – ask the user
  + complete web output to file in report form (for humans), ordered alphabetically by person
  + complete web to file data input form (for later program use), ordering doesn’t matter
  + report on one person – references given and received
  + all highly recommended persons
  + all references written about X, clustered by type (all highly recommended, then all recommended, then all not recommended)
  + all references written by X (same ordering)
  + top X candidates (scoring: recommended = one point, highly recommended = 3 points, not recommended = -3 points)
  + statistics: number of persons, # of references broken down by type
* updates
  + add a person (no associated references)
  + delete a person and all associated references
  + add a reference
  + delete a reference
  + change the level of a reference
    - levels: highly recommended, recommended, not recommended

Additional Ideas for Operations – add if time and inclination allows

* prune the graph by removing all people with a score < X
* more statistics
* quality of a reference, version 1: a reference from a person who himself is highly ranked is worth more
* quality of a reference, version 2: what is the relationship of the person who wrote the reference? a reference from a boos is worth more than a reference from a friend
* multi-level references, example: get all people whom X has given a (good) reference to, plus all people referenced by that list
* notify a person if they have a low score
* reciprocal references: for one person, notify them if they have written a reference to someone who has not written them a reference (the purpose is to prompt that person to write a reciprocal reference)
* notify a person if they have very few references (the purpose is for them to attempt to get more references)
* other ideas you have … must be explained in the cover letter

User interface

* menu or GUI buttons, etc. with all operations
* user enters input and output file names, and query data (e.g., person’s name)

Errors handled

* web file format/syntax: create error log file, do not process
  + input file errors: unknown operation, wrong # of “operands”, person not in web, etc.
* self reference (a person writes a letter for himself), do not add
* user input
  + person not found – ask if they should be added
  + redundant entry – disallow, or offer reasonable alternative
* user tries to exit without saving, prompt before ending program

Input file – use this exact format

* <operation> <”operands”>

Examples

* add person Sam
* add reference Sam Joe highly recommend *from Sam, about Joe*
* change reference Joe Sam do not recommend *Joe does not recommend Sam*
* delete reference Charley Bob *Charley no longer rec.’s Bob*
* delete person Fran

Notes

* actual letters of reference are not stored
* tolerate case differences
* I will want to see a demo of your web, before your final submission; if it fails some random testing, you will need to work on it further
* all discussions should be non-trivial; this is not primarily a programming project, although that is the central portion

Due date: Wed. October 16 (3 weeks)

Extra credit

* with a classmate who is also finished, do a code walkthrough, in which the author presents the code in detail, and the reviewer asks questions about the logic involved

Hand in and post to Moodle

* source code
* sample non-trivial input file
* sample non-trivial output file, data form
* sample non-trivial output file, report form
* sample non-trivial error log
* a diagram of a non-trivial sample test web you used during development
* cover letter with discussions
  + explain your selected data structure(s) in detail (you may have to investigate built-in data structures you are using)
  + explain any additional operations you implemented
  + explain any design decisions or specification clarifications you made during development
  + state and explain the big-O of each of the operations (including any additional ones you implement), along with best case, worst case, and average; do not use an empty web for any of these cases
  + how the code walkthrough went, if you did one

Further notes from class discussion.

* explain why your data structure choice is good for this application
* for the demo, have a sample input file ready with a matching printed sample web for me to refer to during the demo
* you may change the programming language you used on your original graphs package
* outside APIs are OK, as long as they don’t handle major portions of the problem
  + document what you use, in the cover letter
* any extra credit must be excellent to receive credit
* keep a development log: time spent, by day, with activity specified; example:

10/4 1.0 hr. review graphs project

10/4 0.7 hr. create sample webs by hand for later testing

10/5 2.5 hr. translate graphs project to new language

:

* + estimate times for work already done, if you didn’t keep track