Open Source Tools Assignment 3

Due: November 6th, 2014 at 11:59PM

Shell Scripting

For this assignment, we'll be creating a new command called `question` that can be used to ask questions, provide answers, and vote on questions and answers. Think of it as a simplified command line version of Quora. The command will store all of its data in a subdirectory of your home directory.

The format of the command is:

```
question option [args]
```

where *option* is one of the option names described in this assignment (*create*, *answer*, *list*, *vote*, *view*). Each option will have its own set of arguments that follow the option name.

Detailed Description

Each question has its own identifier, which is just a string. The string will have the format *login/name*, where *login* will be the author of the question and *name* is a unique name for the question that is allowed to contain any characters other than a forward slash. A question consists of free text that can contain any characters. There are two constraints: a question cannot be empty, and a question cannot contain the sequence of characters `====` as those will have special meaning later on. Each question has an associated set of answers that can come from any user. To create a question, the *create* option should be used:

```
question create name [question]
```

This command sets up a directory for a new question as question id *$USER/name* (see file structure below). If the question id already exists, an error message should be printed to *stderr* and the command should exit with a non-zero exit status. If the question is invalid, an error should be displayed (again stderr exiting non-zero). If it succeeds, nothing should be printed and the script should exit with status 0. The last argument is optional. If supplied, the question will come from this argument. Otherwise, your script will read from *stdin*.

To answer question, the *answer* option is used. The syntax is as follows:

```
question answer question_id name [answer]
```

The first argument will be the question identifier, which is of the format *login/name* as discussed above. Second is a name for the answer which has the same naming restrictions as questions. Last is an optional answer, that has the same semantics as question in the command above (namely, it must be non-empty, not contain `====`, and comes from *stdin* if the argument is not specified). Running this command will create a file for the answer to this question. If successful, it will exit 0. Your script will create the proper directories and files to store the answer. It should check that the question id specified exists, and the answer id does not, and exit with an error appropriately.

The *list* option can be used to see which questions exist, either globally or created by a specific user. The syntax is as follows:

```
question list [user]
```

The user is optional. When not specified, question ids from all users are shown. The file `/home/unixtool/data/question/users` has a list of all users. When a user is specified, a list of question ids for the given user should be shown, one id per line.

This brings us to voting. To vote in a survey, the *vote* option is used which has the following syntax:

```
question vote up|down question_id [answer_id]
```

Here, the first argument is either up or down depending on whether the vote is positive or negative. *question_id* is required. If *answer_id* is specified, the vote will apply to an answer, otherwise it applies to the question. If the question or answer does not exist, your program should report an error appropriately. The vote itself will get appended to a file with the contents "up" or "down" (see details below). A successful vote should print nothing and exit successfully.

Finally, to see a question with its answers and votes, the *view* option should be used. It is used as follows:

```
question view question_id ...
```

This will print the question and each answer. Before each question or answer is printed, a line with a number is printed. This number will be the total number of "up" votes minus the total number of "down" votes for this question or answer aggregated across all users. Between each question or answer, a line with the delimiter `====` should be printed. The command should first check that the question exists (exiting appropriately if not).
For this command, multiple question_ids can be specified, which will have the same effect as showing each individual question and concatenating the results.
NOTE: You can optionally display the answer_id after the number of votes for each answer. This can help debugging.

Example Usage

Here is an example using the survey command that demonstrates all of the options described above:

```
$ question create q1 "What pet would you like to own?"
$ question answer kornj/q1 a1
Pig
$ question answer kornj/q1 a2 "Dog"
$ question answer kornj/q3 a1
no such question: kornj/q3
$ question list
kornj/q1
lf1238/q7
$ question list dmr
$ question vote up kornj/q1
$ question vote down lf1238/q7
$ question vote up kornj/q1 kornj/a1
$ question vote down kornj/q1 kornj/a2
$ question view kornj/q1
1
What pet would you like to own?
====
1 kornj/a1
Pig
====
```

```
-1 kornj/a2
Dog
```

Implementation

You may use either **sh**, **bash**, or **ksh** to implement this assignment.

You will need to store the information somewhere. You ***must*** use the following directory structure:

- The command should use a root directory called `~/.question` to store all files. Each time invoked, the directory should be created if it does not exist. The permissions of this directory and all files below should be world readable. Make sure your home directory has at least execute permission for all users.
- Within the directory `~/.question`, create three subdirectories called: `questions`, `answers`, and `votes`.
- Within the `questions` subdirectory, there should be a file with the question name where the contents of the file are the question.
- Within the `answers` subdirectory, there should be a subdirectory for the question user, and within that a subdirectory for the question name, and within that a file for the answer name. The contents of this file are the answer for that question.
- Within the `votes` subdirectory, it should look just like the answers subdirectory, except instead of having a subdirectory for the question name, there will be a file. The file contents will consist of all the votes for any questions or answers to the questions. The format of each line of the file will be either:
  vote
  or
  vote answer_id
  where *vote* is either the text `up` or `down`. When a vote is saved, it should be appended to this file. Note that when you read this file, you should account for the fact that a user could vote twice and only use the last entry in this case.

You can look at `/home/kornj/.question` for an example directory corresponding to the example listed above.

Also make sure that your commands handle error conditions and make no assumptions about existing directory structure (it creates whatever directories it needs to if they do not exist). Print a usage message in the following situations:

- No option is given
- No such option exists
- The wrong number of arguments are given to an option (too few or too many)
- Invalid arguments

Always return a non-zero status code when an error condition is encountered.

Turning in the assignment

Turn in a single file that contains the shell script for `question`. You must submit an ASCII text file, **not** a Microsoft Word or pdf file.
Make sure each command is tested on *access.cims.nyu.edu*.

When you finish, submit using the homework submission system.