Programming Assignment 3

Assigned: March 27
Due: April 17

In this assignment, you will implement a toy system of folders (directories) and commands over them.

This assignment should be submitted to Rahul Mangwani manghwani7@gmail.com.

The system of folders at an time forms a tree. Initially, there is a single folder ROOT, which is always the root of the tree. At any given time there is a working folder which is, so to speak, “where you are” in the folder hierarchy. Commands are interpreted relative to the working folder.

Every folder has a name, which is a string of alphanumeric characters. All the subfolders of a particular folder must have different names; however, subfolders in different parts of the hierarchy may have the same name. A folder also has a size.

As in assignments 1 and 2, the input consists of a sequence of commands, one command per line. The system accepts the following commands.

- **C** ⟨name⟩. Create a subfolder of the specified name within the working folder.
- **D** ⟨name⟩. Go down a level to the subfolder of the specified name; that is, change the working folder to be that subfolder.
- **U** ⟨name⟩. Go up a level in the hierarchy; that is, change the working folder to the the parent of the current working folder.
- **R** ⟨name⟩. Remove the subfolder of the specified name. This will indirectly cause the entire subtree associated with that subfolder to be removed.
- **L**. Print all the subfolders of the current working folder, in alphabetical order, separated by a space.
- **P**. Print the path from the root to the current working folder, with the names of the folders separated by slashes.
- **N**. Print the number of folders in the subtree of the current working folder.

**Example**

For instance, suppose that the input begins as follows:

Input
  C Music
  C CompSci
  C History
  D CompSci
  C Assignments
  C Exam
  U
  D History
  C TermPaper
  C Exam
At this point the folder hierarchy looks like this:

Continuing, we execute the following commands

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>ROOT/History/TermPaper</td>
</tr>
<tr>
<td>L</td>
<td>Sources</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
</tr>
<tr>
<td>U</td>
<td></td>
</tr>
</tbody>
</table>
At this point, having executed the "Remove" command, the hierarchy looks like this:

More commands:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>CompSci Music</td>
</tr>
<tr>
<td>N</td>
<td>5</td>
</tr>
</tbody>
</table>

**Data Structures**

Define a class `Folder`. A `Folder` should have the following data fields:

- The name, a `String`. 
• A sorted linked list of the children of \( f \). Use the `compareTo` method on `Strings` to alphabetize. You do not have to use a more sophisticated data structure to maintain the alphabetical order.

• The parent of \( f \).

• The next sibling of \( f \).

• The number of nodes in the subtree underneath \( f \) (including \( f \) itself. Thus, the ‘N’ command can be executed just by retrieving the value of that field.

The current working folder is held in a variable.

**Executing commands**

To execute the command \( C \langle \text{name} \rangle \):

• Check that there is not already a subfolder of the working folder with the specified name. If there is, print an error message.

• Create a folder of the specified name, of subtree size 1.

• Add it in alphabetical order to the children of the working folder.

• Working upward from the working folder to the root, increment the subtree size on each of the ancestors of the new folder by 1.

To execute the command \( D \langle \text{name} \rangle \). Find the child of the working folder of the specified name. If none, give an error message. Change the working folder to be that subfolder. To execute the command \( U \langle \text{name} \rangle \). If the current working folder is the root, print an error message. Change working folder to the the parent of the current working folder. To execute the command \( R \langle \text{name} \rangle \). Find the child \( c \) of the current working folder with the specified name. If none, print an error message. Delete \( c \) from the list of subfolders. Decrement the subtree count at the working folder and all its ancestors by the count of \( c \). To execute the command \( P \). Work upward from the current working folder to the root, then print in backward order (this is easily done with a recursive routine).

L and N are obvious.

**Input/Output**

As in assignments 1 and 2 you may assume that the input is correctly formatted.

**Submission**

Send an email to the grader with the source code as an attachment or multiple attachments.
Honors assignment

Implement two additional commands for changing the current working folder using a path rather than a single name:

- ‘A’ ⟨name-1⟩ ⟨name-2⟩ . . . ⟨name-k⟩ changes the current working folder to a folder specified in terms of an “absolute” path starting from the root. The first name name-1 will always be ‘ROOT’.
- ‘G’ ⟨name-1⟩ ⟨name-2⟩ . . . ⟨name-k⟩ using a relative path starting at the current working folder. The keyword “up” is used to indicate the parent folder. (You may assume that there are no folders named “up”.)

For instance, if you are in the state of the first picture above, and you wish to change the working folder to the Exam subfolder of History you can execute either the command

A ROOT History Exam

or

G up Exam

In the latter ‘up’ takes you up to History and then you go down to Exam.