CS 164, Spring 2009 **CS 164: Homework #2** **P. N. Hilfinger**

Due: Friday, 6 February 2009 at 2400

The main purpose of these exercises is to get you familiar with both Python, which we'll be working on this semester, and the Subversion software. To complete this homework, you must register a team.

It's *really important* that these tools work for you. Be prepared to RTFM. Do *not* just accept error messages as if they were Acts of Nature beyond your control; *whatever it takes, make it work!*

1. If you have not already done so, check out a copy of the Subversion repository directories you can access. The command-line version should look like this:

```
svn checkout svn+ssh://cs164-ta@quasar.cs.berkeley.edu/LOGIN/trunk DIR1
svn checkout svn+ssh://cs164-ta@quasar.cs.berkeley.edu/TEAM/trunk DIR2
```

where *LOGIN* is your login, *TEAM* is the name of your team, and *DIR1* and *DIR2* are names of two local directories you intend to use to hold your personal copies of these repository directories. On the instructional machines, you can use the shorthand

```
svn checkout $REPOS/trunk DIR1
svn checkout $TEAMREPOS/trunk DIR2
```

2. Change (`cd`) to the new working directory *DIR1* and create an empty subdirectory named `hw2`, using the command

```
svn mkdir hw2
```

Create an empty file named `count.py` in this directory. Now tell Subversion about it:

```
svn add hw2/count.py
```

Until indicated otherwise, all the `svn` commands in the rest of the exercises below are supposed to be executed in this same (*DIR1*) directory.

3. You can list files that the repository has stored in it with the command

```
svn ls $REPOS/trunk
```

Why isn't `hw2` there? Didn't you just `svn mkdir` it? (This is a "thought question." Just make sure you know the answer.)

4. Now commit the contents of the *login* directory. Try the commands

```
svn ls $REPOS/trunk
svn ls -R $REPOS/trunk
```

If you actually committed your files, you now should see `hw2/count.py`.

5. Now remove the `hw2` directory you created with the following sequence:

```
svn remove hw2
```

and commit this change. Create the directory anew, but this time from our template:

```
svn copy svn+ssh://cs164-ta@quasar.cs.berkeley.edu/staff/hw2 .
```

(that's a period at the end, preceded by space, meaning "the current directory") or with the shorthand version:

```
svn copy $STAFF/hw2 .
```

Again, commit it.

6. Fill in `hw2/count.py` with a Python program that reads text from the standard input, keeping track of the number of times each word in the text appears, and then prints out the ten most frequently occurring words that are at least 4 letters long, one per line, in order of decreasing frequency. A "word" here is a contiguous substring of letters; anything other than a letter delimits a word. Ignore case (and print words in lower case). Commit your program to the repository.

7. (Make sure you successfully committed your solution before doing this exercise!) Now erase the program you just spent all this time writing with

```
rm hw2/count.py
```

and restore it with

```
svn revert hw2/count.py
```

8. Now get remove the whole `hw2` from the repository with

```
svn remove hw2
```

and commit the change. Make sure it's gone from the repository with the `ls` commands above.

9. Disaster! As a result of the last exercise, you have removed your homework. Now bring it back. First, when you committed the removal of your homework in the previous exercise, `svn` gave you a message such as "`Committed revision` N`.`" That means that your homework was still present in revision $N_1 = N - 1$. If you had deleted your homework some time ago (so that this message was no longer available), you could find out when you deleted it by going to *DIR1* and executing the command

```
svn log -v
```

Try that now. Then retrieve your lost homework with

```
svn copy $REPOS/trunk/hw2@N1 hw2
```

(again in *DIR1*), but *don't* commit this change yet. Make sure you've actually recovered all the files.

10. Now "unrecover" your files with the command

```
svn revert hw2
```

The directory `hw2` will still be there, but it will no longer be under version control. Execute the command

```
svn status
```

and you will see how this fact is reported. At this point, try the command

```
svn commit
```

Nothing should happen.

11. Let's try that again. Remove the directory `hw2` with the command

```
"rm" -rf hw2
```

(use bare Unix commands like this *only* when the thing you're deleting is not under version control.) Now restore it as before and then commit:

```
svn copy $REPOS/trunk/hw2@N1 hw2
svn status
svn commit
```

Without a '`-m`' switch, this should start up an editor for you to enter a log message. Do so. Make sure that the new version of `hw2` is actually committed.

12. So far, you've modified the repository by making changes to your working directory and then committing. Let's work directly on the repository. Create a new copy of your homework in the repository only:

```
svn copy $REPOS/trunk/hw2 $REPOS/trunk/hw2a -m "Extra copy"
```

Now look at your directory with `ls` and with `svn ls`. You will see that `hw2a` is not present. This is because you have not updated your working directory. Try the command

```
svn info
```

and see how it tells you that the revision number of the working directory is not the one reported when you copied. Use

```
svn update
```

and directory `hw2a` should appear.

13. Turn in assignment `hw2` by making a copy in the tags directory:

```
svn copy $REPOS/{trunk/hw2,tags/hw2-1} -m "Handing in hw2"
```

14. With your partner(s), try the following steps inside *DIR2* (where you checked out your team repository):

- Have one of you (let's say Piotr) create a directory `hw2` in *DIR2* with

  ```
  svn copy $REPOS/trunk/hw2 hw2
  svn commit
  ```

- Another of you (let's say Jackie) should do an `svn update` in your own *DIR2*, modify the file `hw2/count.py`, and commit the result.
- Now Piotr should modify his `hw2/count.py` *without* first doing an update. Put the modification at a point in the file about 4 lines from what Jackie modified.
- Piotr should try to `svn commit`. The operation should fail, saying that Piotr's directory is not up to date. Piotr should now do an `svn update`. Assuming that he did this correctly (and made a non-overlapping change), the update should succeed, and Piotr should see both changes in the file. Piotr should now `svn commit`.
- Jackie should now modify her version of `count.py`, but change the same line that Piotr just did (in a different way). Jackie should now try `svn update`. Subversion should report a conflict.
- Jackie should edit the `count.py` and resolve the conflict. Try to commit. This should fail with a message telling you that there are unresolved conflicts. Use either the command

  ```
  svn resolved hw2/count.py
  ```

  or the (more modern) command

  ```
  svn resolve --accept=working hw2/count.py
  ```

  to tell SVN that the conflict is resolved. Now `svn commit` should work for Jackie.
- Finally, Piotr should do the same thing.