CS 164, Fall 2006 **CS 164: Homework #1** **P. N. Hilfinger**

Due: Tuesday, 26 January 2010, 2400

General instructions about homework. Please submit this homework electronically. See the on the class web page. You will need Subversion and you will need to have electronically registered. In any case, please notify us of problems.

You'll find templates for some solutions in the directory `~cs164/hw/hw1` and in the repository at

`svn+ssh://cs164-ta@torus.cs.berkeley.edu/staff/hw1`

or more succinctly on the instructional machines:

`$STAFFREPOS/hw1`

Place answers to any questions that don't require programs in a file called **hw1.txt**.

If you are working at home using your own installation (as opposed to using the instructional machines remotely), you'll need a Python installation. See the Python link from the class home page if you need one.

1. A *subsequence* of a string S (or any kind of sequence, really) is simply a sequence consisting of zero or more characters from S in the same order as in S; for example, `"ack"`, `""`, `"bk"`, and `"back"` are all subsequences of `"back"`, but `"kb"` and `"bb"` are not. A *substring* of S is a *contiguous* subsequence of zero or more characters of S; for example `"ack"`, `"ba"`, `""` and `"back"` are substrings of `"back"`, but `"bk"` is not. Given a string S of length N, how many subsequences does it have? How many substrings? To simplify your life, you *may* assume that letters in S are not repeated (the problem becomes rather "interesting" if you count only distinct subsequences when S may contain repetitions).

2. [From Aho, Sethi, Ullman] Give as simple a description as possible of the languages denoted by the following regular expressions. For example, it is better to describe the language denoted by `((a*b*)*(b*a*)*)*` as "The set of all strings from the alphabet $\{a, b\}$" rather than "A sequence of 0 or more a's followed by 0 or more b's, all repeated 0 or more times and then followed by *etc.*" The latter might be correct, but shows no thought.

a. `0(0|1)*0`

b. `((ϵ|0)1*)*`

c. `(0|1)*0(0|1)(0|1)`

d. `0*10*10*10*`

e. `(00|11)*((01|10)(00|11)*(01|10)(00|11)*)*`

More problems on next page.

3. [Adapted from Aho, Sethi, Ullman] Write the simplest regular expression you can for each of the following languages, using basic regular expressions from Python. Turn in files called **P3a.py**, **P3b.py**, etc., each containing that contain solutions to these problems, using the templates for these files provided in **~cs164/hw/hw1** on the instructional machines. Python "regular expressions" are considerably more powerful than the "classical" basic expressions we want here. For this problem, restrict yourself to using just ordinary characters (that stand for themselves), plus the special characters

```
[ ]  ( )  *  +  |  .  $  ^ ?
```

In the following, "letters" mean lower-case letters. Each pattern that you write must match the *entire* input string (an entire line with the frameworks provided) for a match to succeed, not just an initial segment.

a. Strings of letters that contain all five vowels in order (but not necessarily continguously), each one exactly once.

b. Strings composed of letters a–f in which the letters are in alphabetical order (not all letters have to appear in a string, but those that do must be in order).

c. Comments consisting of a string starting with `/*` and ending with `*/` without any `*/` in between unless it occurs inside balancing quotes `"` and `"`. Quotes must be balanced, so that `/*"*/` is illegal, and only balancing pairs of quotes "deactivate" the `*/` symbol, so that `/*""*/""*/` is illegal (the `*/` occurs between quotes, but they don't balance each other).

d. All strings of 0's and 1's with an even number of 0's and an odd number of 1's.

e. All strings of 0's and 1's that do not contain the substring '011'.

f. All strings of 0's and 1's that do not contain the subsequence '011'.

4. This ends the homework to be handed in. Follow the directions for submitting your homework. The rest of the exercises here are intended to get you familiar with Subversion, or at least the command-line client program `svn`. Handed in or not, it's *really important* that this tool works for you. Be prepared to RTFM (specifically, the *Using Subversion* link on the home page and possibly the *Subversion Manual* as well). Do *not* just accept error messages as if they were Acts of Nature beyond your control; *whatever it takes, make it work!*

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

```
$ svn ls $MYREPOS
```

You should see `hw1` and `tags`. If you don't, find out why not!

6. Now create a new version of `hw1` from your old one, giving it a new name with

```
$ cd ~/mywork
$ svn copy $MYREPOS/hw1 hwx
```

and try `svn ls $MYREPOS` again. Why isn't `hwx` included in this listing? Didn't you just `svn copy` into it? Like the rest of the questions in this assignment, this is a "thought question." Just make sure you know the answer.

7. Try the command

```
$ cd ~/mywork
$ svn status
```

You should see an indication the `hwx` is not yet committed. Create a new empty file in `hwx` with

```
$ touch hwx/hwx.txt
```

and again execute `svn status`. This time, you should see `hwx.txt` marked with a '`?`' to tell you that you haven't told `svn` what to do with it. Use the command

```
svn commit hwx
```

to commit your new directory.

8. Try `svn status` again. You should see that `hwx.txt` is still listed as '`?`'. Use `svn add` to inform `svn` about `hwx.txt`. Now the status will have it marked '`A`', but `svn ls $MYREPOS/hwx` will not list it. Commit `hwx`, and make sure that `hwx/hwx.txt` is properly part of the repository.

9. Now remove `hwx/hw1.txt` *without* telling `svn`:

```
$ rm ~/mywork/hwx/hw1.txt
```

and see what `svn status` and `svn ls $MYREPOS/hwx` show. In the documentation we've provided for Subversion, you will find a command for recovering `hw1.txt`. Find it and use it.

10. Erase the `hwx` directory in the repository with

```
svn remove $MYREPOS/hwx -m "Remove hwx"
```

Why is `/mywork/hwx` still around? Why doesn't `svn status` show it as deleted? Execute

```
$ cd ~/mywork
$ svn update
```

This should get remove the directory.

11. Assuming that you submitted your hw1 solution as directed, the `svn update` from the last problem will have fetched all your `tags/hw1-`*N* submission tags. This is rather annoying, because in general, you won't need working copies of these tags (except temporarily to check them after submission, perhaps). You can tell `svn` that in the future `svn update` should not fetch working copies of these tags with the command

```
$ cd ~/mywork
$ svn update --set-depth empty tags
```

12. Wait: you mean you actually deleted `hwx` from the repository? Disaster! You'd better bring it back. First, look for where this happened, using

```
$ svn log $MYREPOS
```

or for more information,

```
$ svn log -v $MYREPOS
```

This will show you all your commits, and let you find out the revision in which you deleted `hwx`. Now restore it from the revision before this happened. If `svn log` tells you that you deleted `hwx` in version 1000, then execute

```
$ cd ~/mywork
$ svn copy $MYREPOS/hwx@999 .
$ svn commit
```

to restore and commit `p1`.