Homework #2: Install Subversion and set up course Subversion Repository

To successfully complete the homework assignment

1. **check out the tools repository**
2. **create a personal branch repository for future use**
3. **Add a file called foo.txt to your branch**

Read the following to help you complete the above tasks.

NOTE--We will use shorthand for some information. We will use "make" variable syntax to express some strings in commands. In particular, our Subversion repository address will be expressed as if the following variable assignment had been done,

SVN=https://svn/cs.georgetown.edu/svn/projects2

Use the path above or the one that was announced in class. When we want to refer to that string, we will us "make" variable substitution syntax, e.g., ${SVN} will substitute for https://svn/cs.georgetown.edu/svn/projects2 .

Other variable/string pairs we will use are,

COURSE-YEAR=120-2016

where the course number and year match the course you are enrolled in,
and

AUTH=--username 'hw-sys-arch' --password 'y(&qwqsq'

where the username and password were announced in class or try the ones above.

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Before installation, if you need a quick referesher of Unix commands or must first install a Unix-like environment, see the following resources (use login info above).

URL = https://svn/cs.georgetown.edu/svn/projects2

---- Using a web browser, see the READMEs and documentation in,
URL/tools/Editors_Unix
URL/tools/Subversion

-- MS Windows, install cygwin. Take all defaults to get a minimal installation. Check that it runs. Then rerun setup.exe to add other tools to your installation. Be sure to include an editor (vi or emacs). When you run setup, you will see a list with items such as "Devel". If you click on the "+" for one of these, you will see a bunch of tools you could add to your cygwin installation. Clicking on "Skip" will select the item to add.

-- OSX, use your terminal app, find your home directory, try some unix commands. Install XCode if you haven't already.

@@

====== Subversion (SVN) Client Installation ==============
Subversion consists of **two parts, a server, and a client**. You only need a client. Most downloads will include a server, but **you do not need to set up the server**. Is a **command-line client svn** already installed in your system? Try "svn" in a terminal window.

If it is installed, you should get an error message about too few arguments. Even if you get a "command not found" error message, svn might be installed but your PATH environment variable does not include the path to it.

You can use,
find / -name "svn"

to search for it. (That can take a long time.)

If not, **is an executable binary available?** (Rather than downloading source code and building.)

--- **Mac OSX** 10.5 and later: use the terminal app.
Get XCode (older ones are free), see Apple Developer Connection.
(See MacPorts.org to download binaries not included in XCode, if needed.)
(See this link for a discussion of three OSX unix package facilities (Fink, MacPorts, HomeBrew)

http://apple.stackexchange.com/questions/32724/what-are-pros-and-cons-for-macports-fink-andhomebrew

--- MS **Windows: Do not install binaries for gui svn clients** from the Subversion web site or
elsewhere.
You need a **unix interface to Windows** anyway; so, **install cygwin**:
http://www.cygwin.com/
setup.exe ===> Lots of **selections** you can make
--- Base: gzip, grep, sed, tar, which

--- Devel: gdb, make, subversion
--- Editors: emacs, vim
--- Net: openssl

First install: **take all defaults**
rerun later: **select things to add**

============= Checkout CourseDocuments and Create Your Branch

You have to have your shell's editor environment variable set to your favorite editor. See **Subversion Commits (checkins)** below. Now that you have command-line svn available, use svn to "svn checkout" our class tools. First, cd to a convenient place in your local machine's file system where you want a working copy of CourseDocuments to be installed. Then,

```
svn co ${SVN}/${COURSE-YEAR}/CourseDocuments ${AUTH}
```

The advantage of using the above is that you can put the text into a file, say, "foo.txt", and have the shell execute the file for you this way,

```
bash foo.txt
```

NEVER do SVN IMPORT or SVN EXPORT.

Get your own local "working" copy so you can see updated documents:

```
svn co https://svn.cs.georgetown.edu/svn/projects2/tools/
svn co https://svn.cs.georgetown.edu/svn/projects2/121-2017/branches
```

Local names of your working copies will be "**tools**" and "**branches**"
in your system's directory tree where you did **svn co**.

**************Do not commit changes to these**.

To get latest versions of the tools:

```
cd tools
svn up
```

To Create your branch:

```
svn co ${SVN}/${COURSE-YEAR}/branches/
cd branches
```

svn mkdir JeremyBolton (No, silly, use your own name)
svn status
svn ci
Your branch is for your work. After this, **ONLY commit changes ("svn ci") to your branch**.

Now delete "branches" and check out just your branch:
cd ..
/bin/rm -rf branches
svn co ${SVN}/${COURSE-YEAR}/branches/JeremyBolton (No, silly, use your own named directory)

Locally renaming your working copy is ok, but **only at the root of a working copy**:

mv JeremyBolton MyBranch

This has no effect on the contents of the repository. For **help** with Subversion commands,

svn help

Sometimes you will mess up your working copy. You will not want to check in the mistakes. Instead, move the files you made changes to to a safe place, then clobber your working copy:

/bin/rm -rf JeremyBolton

=========== Using Subversion
---- Now that you have a working copy of your branch, you can add something to it. First, "cd" to your branch,

cd <your-branch>

Next, create a plain text file, e.g., foo.txt, using a command-line editor. Add a few characters to the file, then save it.

Now, check to see what Subversion thinks the status of your branch is at this point,

svn status

The responses you can get are, "?" (svn doesn't know about the file or directory), "A" (svn will add it to the repository at the next commit), "D" (svn will delete it at the next commit), ... and so forth.
---- Next, schedule the file to be added to your branch,

svn add foo.txt

and do another "svn status" to see the change. The repository has not changed at this point. You could, using the shell, delete your entire working copy of your branch, and nothing will ever change in the repository.

---- Now commit the change, i.e., send the changes to the repository,

svn ci

Make sure you add a commit log comment. If you did not yet set up your shell environment variable (VISUAL or EDITOR or SVN_EDITOR, whichever works for you svn version), you need to do that before you make the commit.

---- Next, run **svn log -v** to see the changes that have been made to this section of the repository. You will find that nothing has changed, according to the log. That is because the log was changed in the repository when you committed. You need to download the changes to the log. Run **svn update** and then run **svn log -v** again.

You should now see your log comment and the files that changed in the commit.

Locally renaming your working copy is ok, but **only at the root of a working copy**:

mv JeremyBolton MyBranch

This has no effect on the contents of the repository. For **help** with Subversion commands,

svn help

Sometimes you will mess up your working copy. You will not want to check in the mistakes. Instead, move the files you made changes to to a safe place, then clobber your working copy:

/bin/rm -rf JeremyBolton

=========== Subversion Commits (checkins)
Your unix environment variables (PATH, VISUAL) need to be set.
They are set by your shell when it starts, by reading commands from a file.

You need to add commands to this file (assuming you use bash):
~/.bash_profile
(or another, such as ~/.bashrc or ~/.profile)
("~" is a synonym for your unix home directory,
e.g. /home/bolton)

You need a command-line editor (**VI or EMACS**).
Commits require a log comment. You must edit the comment as part of the commit process. svn starts an editing session for you. When you write/save the comment, the commit takes place.

Set your editor environment variable, e.g.:
export VISUAL=vi
because svn needs to know which editor to execute. Different svn versions look for different environment
variables (e.g., VISUAL or EDITOR or SVN_EDITOR.)

Set your PATH variable, e.g.:
PATH="/home/bolton/bin:${PATH}"
(Of course, this depends on what executables you have installed and where you put them.) When you do this in a command-line shell,

```
>>> ls
```

each path prefix in PATH, such as "/home/bolton/bin", is tried in turn to see if the executable exists:

/home/bolton/bin/ls

If not, the next prefix in PATH is tried.
See all your shell's variables using the shell command, **set**.