24-783 Problem Set 1

Due: 1-week assignment. Please see Canvas for the exact due date.

(*) In the instruction (and in all of the course materials), substitute your Andrew ID for when you see a keyword *yourAndrewId*.

In this problem set, you set up programming tools called SubVersion, Git, and CMake. You also practice using the command line, checking in and out the source repository.

PS1-1

Follow the instruction of the course note and install SubVersion and Git clients and CMake on your computer. Read the software instruction and make it available from the command line (Terminal in macOS and Linux. PowerShell or CommandPrompt in Windows.)

PS1-2

Meet with a TA during the office hour and create a SubVersion account.

PS1-3

Do the following using the command line.

Create a directory called 24783 under your home directory ("~" in Linux, macOS, and Windows PowerShell. %USERPROFILE% in Windows Command Prompt.) You can choose to keep this directory in a different location, but the rest of the instruction assumes that you created it directly under your home directory. Interpret the instruction accordingly.

From now on, I use "~" for your home directory. I use "/" (forward slash) as a directory separator, but in Windows command prompt, you need to replace it with "\" (back slash).

Then create a directory called "src" under 24783 directory you've just created.

Change directory to ~/24783/src and git-clone the following repository:

    https://github.com/captainys/public.git

and svn-checkout the following repositories.

https://ramennoodle.me.cmu.edu/svn/teaching/24783_S19/students/*yourAndrewId*

https://ramennoodle.me.cmu.edu/svn/teaching/24783_S19/course_files

If you check out all files correctly, your 24783_S19 directory tree will look like (Only showing up to the third-level):

- src
  - course_files
    - ps1
  - *yourAndrewId*
  - public
    - Doxygen
    - Experimental
    - iOS
    - src
    - tests
    - UWP
    - CMakeLists.txt

Every week, when you start an assignment, do:

cd ~/24783_S19/src/public

git pull

and

svn update *yourAndrewId*

to keep your files up to date.

PS1-5 Making CMake projects

Find source files for the Cannon-Ball game from 24-780. Change directory to your directory, and type:

```
svn copy ../course_files/ps1 .
```

This directory has source files. Your task is to write three CMakeLists.txt files:

- ~/24783/src/*yourAndrewId*/ps1/CMakeLists.txt
- ~/24783/src/*yourAndrewId*/ps1/simplewindow/CMakeLists.txt
- ~/24783/src/*yourAndrewId*/ps1/cannonball/CMakeLists.txt

The first CMakeLists.txt must add sub-directories, cannonball and simplewindow. File names are case sensitive. It's no longer Windows world.

For practice purposes, you write a CMakeLists.txt file for the simplewindow library instead of using the library in the *public* repository.

The second one is a library. The library name must be "simplewindow". No hyphen. Case sensitive. See the lecture note and cut & paste necessary part. (You won't need "bouncing_ball" executable in this CMakeLists.txt) Not complying with the naming rule will be the basis for automatic 30% penalty.

The last one is an executable (**make sure to add MACOSX_BUNDLE to make it available for macOS environment**) for the Cannon-Ball game. The executable name in the CMakeLists.txt must be "cannonball". No hyphen. Case sensitive. Not complying with the naming rule will be the basis for automatic 30% penalty. This executable must link "simplewindow" library.

Remember, our TAs will grade using a grading script. Not complying with the naming rule will stop the script, and force TAs to manually look into your submission, which will clearly make TAs unhappy. You won't want to be graded by an unhappy TA!

These three files must be added to the SubVersion's control.

Make a build directory somewhere outside ~/24783/src, and run cmake for:

```
~/24783/src/yourAndrewId/ps1
```

and build the project. Make sure you can run the program.

PS1-6 Submitting with SVN

After writing three CMakeLists.txt, and verified that you can compile and run the Cannon-Ball game, change directory to your directory, and then type:

```
svn commit . -m "Submitting Problem Set 1"
```

(Message can be anything that is descriptive.)

You can commit as many times as you want. Actually, we strongly encourage to commit when you make some progress in the assignment. That way you can keep track of the changes.

Then, never skip the verification after submitting the final version by following the steps below.

Verify your submission.

1. Create a directory ~/24783/verify.
2. Change directory to ~/24783/verify.
3. Check out:
   https://ramennoodle.me.cmu.edu/svn/teaching/24783_S19/students/*yourAndrewId*
4. That's what TAs will grade. Make sure you have uploaded all the files, and file names comply with the specification. You are responsible for making sure all files are submitted according to the problem-set specification.

If you do not see some files that you are supposed to upload, the file may not be added to the local repository. Use:

svn add

command to add them and commit.

If you made a mistake in the file name, use:

svn rename

to change the name and then commit. DO NOT rename from the file-explorer or finder or ren-command or mv command. Use "svn rename" so that SubVersion recognizes the change.

Make sure to do those changes for ~/24783/src/*yourAndrewId*, and do not edit files under ~/24783/verify. It is easy to mix up.

If you had to make changes and commit, verify your submission again by change directory to ~/24783/verify/*yourAndrewId*, and then type:

svn update

To see your changes are uploaded to the server.

After verifying, you can delete ~/24783/verify directory.

Oops! You deleted both your ~/24783/src and ~/24783/verify! Not to worry, if you have correctly uploaded (committed) files to the SVN server. Follow the steps of PS1-3 to check out files from the server. That's why you should commit often.

It would be a disaster if you deleted files before committing to the server. So, don't do it.

Rather, it is recommended to frequently check in to the server. The SVN server will keep track of your changes so that you can undo some changes if you mess up your code.

Reminder: Did you add **MACOSX_BUNDLE** keyword? This is the most common mistake since Windows and Linux doesn't care about this keyword. Even I often make a mistake. Double check that you didn't forget MACOSX_BUNDLE in add_executable command.