CSCI 160 student git instructions

The basic idea is as follows and detailed instructions are provided afterwards:

- A central git repository is maintained by the department, with a section within it for each CSCI160 lab section.
- The project will have several repositories containing the relevant material the instructor is distributing to the students.
- Each student will also have a central repositories where they maintain their current official submissions for the project.
- At the start of term the students will initialize their own repositories as a copy of the instructors'.
- The student will then create a working copy of each repository within their regular csci account (e.g. on otter), and use that copy to make all their desired changes as they build their solution and create documentation.
- The student will use appropriate git commands (add, commit, etc) to maintain their local working copy, and as often as desired (prior to the deadline of course) the student can push their most recent version of their work back to the central repository.
- After the lab deadline passes, the instructor will retrieve the latest version of each student's work to use that as the basis for marking the lab.

Specific instructions

Note that the examples below are based on csci160-XX (where XX is your lab section, 01, 02, etc.) and a lab named labX (where X is 0, 1, 2, …11).

Step 1, 2, 3, and 4 are done once only at the start of the term, while steps 5 and 6 will be done once for each lab and assignment.

For submitting labs

1. Once only, at the start of the course, on otter (or one of the pups or cubs) create a directory for your csci160 material (**this year, please ignore this step, it is already done**):

   ```
   mkdir ~/csci160
   ```

2. You can see your current central repository information using the command

   ```
   ssh csci info
   ```

   For each repository you have access to, this shows the name of the repository, and one or more of RWC (for Read access - you can fork/clone it, Write access - you can push to it, Creator - you created it)

3. Create your own fork of the labs and the assignment using the command:

   ```
   ssh csci fork csci160-XX/lab1 csci160-XX/$USER/lab1
   ssh csci fork csci160-XX/lab2 csci160-XX/$USER/lab2
   ssh csci fork csci160-XX/lab3 csci160-XX/$USER/lab3
   ssh csci fork csci160-XX/lab4 csci160-XX/$USER/lab4
   ```

```
ssh csci fork csci160-XX/lab5 csci160-XX/$USER/lab5
ssh csci fork csci160-XX/lab6 csci160-XX/$USER/lab6
ssh csci fork csci160-XX/lab7 csci160-XX/$USER/lab7
ssh csci fork csci160-XX/lab8 csci160-XX/$USER/lab8
ssh csci fork csci160-XX/lab9 csci160-XX/$USER/lab9
ssh csci fork csci160-XX/lab10 csci160-XX/$USER/lab10
ssh csci fork csci160-XX/lab11 csci160-XX/$USER/lab11
ssh csci fork csci160-XX/assign1 csci160-XX/$USER/assign1
```

Note that this creates a repository of your own on the central server containing a copy of the instructor's repository.

4. Now it is time to create working copies in your own account for each lab:

```
cd ~/csci160
git clone csci:csci160-XX/$USER/lab1
git clone csci:csci160-XX/$USER/lab2
git clone csci:csci160-XX/$USER/lab3
git clone csci:csci160-XX/$USER/lab4
git clone csci:csci160-XX/$USER/lab5
git clone csci:csci160-XX/$USER/lab6
git clone csci:csci160-XX/$USER/lab7
git clone csci:csci160-XX/$USER/lab8
git clone csci:csci160-XX/$USER/lab9
git clone csci:csci160-XX/$USER/lab10
git clone csci:csci160-XX/$USER/lab11
git clone csci:csci160-XX/$USER/assign1
```

NOTE: This command will only work if either the directory doesn't exist, or if it exists and is empty. If you've already put files in your lab directory and try to clone, you'll see an error like this:

```
fatal: destination path 'lab3' already exists and is not an empty directory.
```

You must empty the directory (copy the files elsewhere) and then run that clone command again.

5. To submit a weekly lab, do the following:

- Move to the appropriate repository (i.e., labXX)

```
cd labXX
```

- Create the files you are going to submit, and add any content you wish
  - For Sarah's labs, this will include copying the prelab, checklist, and labx.cpp file to this directory
- For each file, add the file to git (this tells git that you want this file to be updated (submitted)) using the following command:

```
git add labXX.cpp

git add LabXXChecklist.pdf
```

```
git add LabXXPrelab.pdf
```

- Commit the files:

```
git commit -m "---submitting lab xx ---"
```

- Push the files:

```
git push
```

6. When/if you need to **update** a file (you made some changes, you pushed the wrong version…)
   - Suppose you changed your labXX.cpp file:

```
git add labXX.cpp

git commit -m " updating lab xx to add comments "

git push
```

NOTE: if you update a file *after* the submission date, please email your instructor to notify them. If they've already grabbed the labs to mark they may need to take extra steps to make sure they have the correct version.

Other advanced commands (only for the brave!!):

```
git rm filename if you want to get rid of a file (-r for directory)
git mv oldname newname if you want to move or rename a file or
directory
git add -u (so any files you've removed locally also get removed from
the repository)
```