CS 458 - Homework 4

Deadlines

Problems 3 onward are due by 11:59 pm on Friday, October 7, 2016

(Problems 1 and 2 were completed during the Week 5 and Week 6 Labs, respectively.)

How to submit

Submit your files for Problem 3 onward for this homework using `~st10/458submit` on **nrs-projects**, with a homework number of `4`

(Problems 1 and 2 were completed during the Week 5 and Week 6 Labs, respectively.)

Purpose

To practice lightly with use cases, to start some project-related tasks, and to play a bit with simple branching and merging in Git.

Problem 1

Your team's files `458lab5-team-meeting.txt` and `458lab5-part2.txt` from the Week 5 Lab, submitted during that lab using `~st10/458submit` with a number of **85**, is "counting" as Problem 1 of this homework.

(That is, you will be graded on working in your team during that lab, and whether your team successfully completed these files, meeting the stated specifications, during that lab.)

Problem 2

Your team's file `458lab06-notes.txt` from the Week 6 Lab, submitted during that lab using `~st10/458submit` with a number of **86**, is "counting" as Problem 2 of this homework.

(That is, you will be graded on working in your team during that lab, and whether your team successfully completed this file, meeting the stated specifications, during that lab.)

Problem 3

CAREFULLY read through the project handout -- especially note the following sections:

- Important project goals
- A Few Important Points Re: Project Teams
- Required Comments for Project Code
- ...and of course all of the project milestones.

Now -- all of the teams should have created a team meeting file during the Week 6 lab. Only ONE of you should have added it to the team repository on GitHub -- but ALL of you can verify that it has been done, and done as required in the project handout in the section "A Few Important Points Re: Project Teams".

On nrs-projects, show that you can clone a copy of your team's private repository from GitHub:

`git clone https://github.com/hsu-cs458-f16/`*`yourTeamGitHubRepo`*`.git`

...and enter your GitHub username and password when prompted.

Verify that your repository has a directory named `team-meetings`, and that there is a file in it with your Week 6 Lab team meeting form, in a file whose name includes the date written as `2016-09-28` (for example, `meeting-2016-09-28.pdf`).

If it doesn't, you need to consult your teammates and make sure these are added to your repository before completing this problem. (Do you have it? See the note at the end of this problem.) How can you get an updated copy of the repository once these have been added?

```
git pull
```

...should do the trick. Now make sure your updated local repository on nrs-projects indeed has the `team-meetings` directory with the Week 6 Lab team meeting form `meeting-2016-09-28.pdf`. When it is there, demonstrate this by running the following commands:

```
# make sure you run the following from the team-meetings directory
#     within your nrs-projects local cloned copy of your team's
#     repository.
echo yourName >> ../../458hw4-show-meeting.txt
pwd >> ../../458hw4-show-meeting.txt
ls >> ../../458hw4-show-meeting.txt
git status >> ../../458hw4-show-meeting.txt
```

Submit your resulting file `458hw4-show-meeting.txt` (remembering to:

```
cd ../..
```

...first, to get to the directory where it is!)

(And I'll also look to verify that your team's repository on GitHub indeed has the `team-meetings` directory with the Week 6 Lab team meeting form `meeting-2016-09-28.pdf`, also.)

[**NOTE**: If your repo doesn't have the `team-meetings` directory with the Week 6 Lab team meeting form `meeting-2016-09-28.pdf`, and you have this file, you should create this directory and add this file! If you do this via nrs-projects, I found that I needed to add the new directory:

```
git add team-meetings
```

...to stage the new directory and its new file -- once the directory has been staged, adding a new file to that directory later only seems to require:

`git add` *newFileName*

...to stage the new file. And, of course, then commit the changes appropriately,

```
git commit -m "made team-meeting and first meeting form"
```

...and push these changes to the GitHub copy of the team's repository:

```
git push origin master
```

]

Problem 4

The following is adapted from the Self-Assessment Exercise 5 on p. 67 of the Jalote course text.

Consider use cases as described in Chapter 3 of the Jalote text.

Also consider the following system: a conference management site which allows authors to submit papers, program chairs to assign reviewers and do the final paper selection based on reviews, and reviewers to enter their reviews.

You should be able to determine several use cases for this system. Select **one** of these that you think would be a main use case.

Start a file `458hw4-use-case.pdf` with your name, and write your selected main use case in the same style seen in Figure 3.4 on p, 53. Be sure to include a primary actor, preconditions if they are applicable, a main success scenario, and exception scenario(s) if applicable.

Submit your resulting file `458hw4-use-case.pdf`.

Problem 5 - stage 1

(some references I used in kluging this together:

- http://pcottle.github.io/learnGitBranching/
- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
- https://git-scm.com/book/en/v2/Git-Branching-Branch-Management
- https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows

)

You are going to use GitHub team `cs458`'s repository `458hw03` for several problems, along with `git` on nrs-projects. The goal is to give you a little taste/experience with Git branching and merging, so that you can hopefully make use of these while working on your team project.

(Remember that if you are prompted by these commands for your password, that should be your **GitHub** password...)

- Make a new `458hw04` directory on nrs-projects, protect it, and change to it:

  ```
  mkdir 458hw04
  chmod 700 458hw04
  cd 458hw04
  ```

- And now, clone the `458hw03` repository that GitHub team `cs458` has access to into this directory on

nrs-projects:

```
git clone https://github.com/hsu-cs458-f16/458hw03.git
```

...and enter your GitHub username and password when prompted.

- Demonstrate that you have reached this stage by running the following commands (STARTING in the nrs-projects directory in which you ran the above command, which should be the one now **containing** your cloned repository directory) (note that >> appends to a file...)

```
echo yourName >> 458hw04-stage1.txt
pwd >> 458hw04-stage1.txt
ls 458hw03/* >> 458hw04-stage1.txt
```

Submit your resulting file `458hw04-stage1.txt`

Problem 5 - stage 2

- Now do:

```
cd 458hw03
```

...so you are in your cloned GitHub repo. And go to subdirectory `team-greetings`:

```
cd team-greetings
```

Basic idea - you can use a branch to explore an idea, or work on adding a particular user story, or work on fixing a particular bug, etc.

But there are two parts of this -- **creating** the branch, and then **checking out** that branch. If you create a new branch and don't check it out, you'll keep working in whatever branch you WERE in -- any changes won't be made in the new branch!

So, commands `git branch` and `git checkout` can do this (yes, there are shorthands, also, BUT we'll do these "longhand" for these problems to hopefully make it clearer what you are doing).

```
git branch myNewBranch  # create a new branch myNewBranch
git checkout myNewBranch # now you have checked out branch myNewBranch,
                         #    and changes you make and stage and commit
                         #    will be part of this branch (until you
                         #    check out another branch)
```

- Create and checkout a branch named `458hw04-branch`:

```
git branch 458hw04-branch
git checkout 458hw04-branch
```

It turns out that you can use the command `git branch` with no arguments to list your current branches -- and the output will show an asterisk `*` next to the current branch.

- Run the following commands to see this, and then to record in a file to submit that you have indeed created and checked out this branch (remembering that, right now, you should be in the `team-greetings` directory of your `458hw03` directory):

```
git branch
echo yourName >> ../../458hw04-stage2.txt
git branch >> ../../458hw04-stage2.txt
```

You're now going to make some specific changes in this branch `458hw04-branch`.

- As part of Homework 3, you created a file in `team-greetings` named *yourGitHubUsername*`.txt` that contained a greeting and your real name.

  Now EDIT this file, **adding** a message saying this is part of Homework 4 - stage 2 and was added in branch `458hw04-branch`. (Leave your Homework 3 contents, do not remove them!)

- Stage and commit your modified file:

  ```
  git add yourGitHubUsername.txt
  git commit -m "doing HW 4 stage 2"
  ```

The command `git branch -v` shows the last commit on each branch.

- Run the following commands to see your current status and last commit on each branch, and also record them for homework problem submission:

  ```
  git status
  git status >> ../../458hw04-stage2.txt
  git branch -v
  git branch -v >> ../../458hw04-stage2.txt
  ```

Submit your resulting file `458hw04-stage2.txt`

Problem 5 - stage 3

Now -- you are "called away" to fix ANOTHER problem, and you need/want to go back to the `master` branch and come back to `458hw04-branch` later.

My understanding thus far is: as long as you commit your changes in your current branch, it should be no problem to switch to/check out another branch, and then come back to this one later.

Conveniently, you just happened to commit your branch `458hw04-branch` at the end of the previous stage.

- So, attempt to switch back to your `master` branch, running:

  ```
  git checkout master
  ```

- Show that you have switched:

  ```
  echo yourName >> ../../458hw04-stage3.txt
  git branch
  git branch >> ../../458hw04-stage3.txt
  ```

- Now -- you are IN the `master` branch. Another way you can tell: look at the contents of the file

yourGitHubUsername`.txt` -- they should, indeed, be the original contents! Run the commands:

```
more yourGitHubUsername.txt
more yourGitHubUsername.txt >> ../../458hw04-stage3.txt
```

- Create and checkout a branch named `458hw04-aside`:

```
git branch 458hw04-aside
git checkout 458hw04-aside
```

- ...and see/show that you have done so:

```
git branch
git branch >> ../../458hw04-stage3.txt
```

- Edit this branch's version of *yourGitHubUsername*`.txt`, carefully adding a message saying this is for Homework 4 - stage 3's aside.
- Stage and commit your modified file:

```
git add yourGitHubUsername.txt
git commit -m "doing HW 4 stage 3"
```

- Run the following commands to see your current status and last commit on each branch, and also record them for homework problem submission:

```
git status
git status >> ../../458hw04-stage3.txt
git branch -v
git branch -v >> ../../458hw04-stage3.txt
```

Submit your resulting file `458hw04-stage3.txt`.

Problem 5 - stage 4

Now you are going to merge the change you just made in branch `458hw04-aside` into your `master` branch.

To do this, you'll switch branches back to your `master` branch, and then use the `git merge` command with the name of the branch whose changes you'd like to attempt to merge with the `master` branch.

- First, go back to the `master` branch, and show that you have done so:

```
git checkout master
echo yourName >> ../../458hw04-stage4.txt
git branch -v
git branch -v >> ../../458hw04-stage4.txt
more yourGitHubUsername.txt
more yourGitHubUsername.txt >> ../../458hw04-stage4.txt
```

- ...and now merge your work from branch `458hw04-aside` into the current `master` branch:

  ```
  git merge 458hw04-aside
  git branch -v
  git branch -v >> ../../458hw04-stage4.txt
  more yourGitHubUsername.txt
  more yourGitHubUsername.txt >> ../../458hw04-stage4.txt
  ```

SINCE you've finished with the `458hw04-aside` branch now, you can delete it, using the `git branch` command's `-d` option.

- Run the commands:

  ```
  git branch -d 458hw04-aside
  git branch -v
  git branch -v >> ../../458hw04-stage4.txt
  ```

Submit your resulting file `458hw04-stage4.txt`.

Problem 5 - stage 5

Now you are going to go back to your branch `458hw04-branch` and continue working on it.

- Run these commands to go back to branch `458hw04-branch` and show that you have done so:

  ```
  echo yourName >> ../../458hw04-stage5.txt
  git checkout 458hw04-branch
  git branch -v
  git branch -v >> ../../458hw04-stage5.txt
  more yourGitHubUsername.txt
  more yourGitHubUsername.txt >> ../../458hw04-stage5.txt
  ```

- And now "complete" your work on this branch, first by adding another line to *yourGitHubUsername*`.txt`, with a message saying you've now FINISHED branch `458hw04-branch`.
- Stage and commit your modified file:

  ```
  git add yourGitHubUsername.txt
  git commit -m "doing HW 4 stage 5"
  ```

- Run the following commands to see your current status and last commit on each branch, and also record them for homework problem submission:

  ```
  git status
  git status >> ../../458hw04-stage5.txt
  git branch -v
  ```

```
git branch -v >> ../../458hw04-stage5.txt
```

Submit your resulting file `458hw04-stage5.txt`.

Problem 5 - stage 6

And, now I decide I'm happy with the work in branch `458hw04-branch`, and want to merge it into my `master` branch.

Again, I'll use `git checkout` to switch to the branch I want to merge INTO, and then use `git merge` with the name of the branch whose changes I want to bring in.

- First, go back to the `master` branch, and show that you have done so:

  ```
  git checkout master
  echo yourName >> ../../458hw04-stage6.txt
  git branch -v
  git branch -v >> ../../458hw04-stage6.txt
  more yourGitHubUsername.txt
  more yourGitHubUsername.txt >> ../../458hw04-stage6.txt
  ```

- ...and now attempt to merge your work from branch `458hw04-branch` into the current `master` branch (which, by the way, **should fail**, complaining about a **merge conflict**.)

  ```
  git merge 458hw04-branch
  git status
  git status >> ../../458hw04-stage6.txt
  ```

- Files with merge conflicts now have additional lines added, called **conflict resolution markers**.

  Check these out, and show you have them:

  ```
  more yourGitHubUsername.txt
  more yourGitHubUsername.txt >> ../../458hw04-stage6.txt
  ```

- For this homework's purposes, it turns out that you want to keep all of the lines added in the various stages, SO:

  Carefully edit your *yourGitHubUsername*`.txt`, removing **JUST** the conflict resolution marker lines:

  ```
  <<<<<<< HEAD
  =======
  >>>>>>> 458hw04-branch
  ```

- After you have resolved each of the sections with conflicts in each conflicted file, you stage its commits to mark it as resolved. You only have one such file and you've handled its conflicts, so do this staging:

  ```
  git add  yourGitHubUsername.txt
  ```

  ...and commit it:

```
git commit -m "doing HW 4 stage 6"
```

- Run the following commands to see your current status and last commit on each branch, and also record them for homework problem submission:

  `more` *yourGitHubUsername*`.txt`

  `more` *yourGitHubUsername*`.txt >> ../../458hw04-stage6.txt`

  ```
  git status
  git status >> ../../458hw04-stage6.txt
  git branch -v
  git branch -v >> ../../458hw04-stage6.txt
  ```

SINCE you've finished with the `458hw04-branch` branch now, you can delete it, using the `git branch` command's `-d` option.

- Run the commands:

  ```
  git branch -d 458hw04-branch
  git branch -v
  git branch -v >> ../../458hw04-stage6.txt
  ```

- Finally, cross your fingers and try pushing your updated `458hw03` repository back up to GitHub:

  ```
  git push origin master
  ```

  (I am hoping very hard that, since each of us should only be changing "our" file in team-greetings, that this MIGHT just work...!)

Submit your resulting file `458hw04-stage6.txt`.