Assignment 6: Collaborative Editing, Revised Version

1 Overview

After reading your blog posts, I can see that many of you have no problem writing understandably. On the other hand, many of the posts would benefit from inspection by another pair or two of eyes. There are poorly constructed sentences, incorrect uses of words, incorrect grammatical constructions, and spelling mistakes that a spell-checker cannot catch because they are valid words. The content of many of your posts is truly outstanding, but the message is weakened by the delivery.

I believe it is very important that your writing is good. People who would look to hire you make judgments based on such things as your ability to speak well and write well. Your ability to communicate in writing and orally is an important part of your skill set. Countless recruiters have said as much to me personally and in their public writings about this.

Therefore, I have devised an assignment for you to do that aims to accomplish three important objectives:

- to give you experience in collaborative work in small teams,
- to give you experience reading other people's writing with an eye towards improving it (i.e., constructive commenting), and
- to develop your *Git* skills further.

In short, you will be assigned to a team of three to four people who will fork and clone each other's blog post repositories, suggest changes, make pull requests to the original owners to improve the posts, and as the owners of the blog post repositories, create remote and local branches that contain the teammates' revisions. This will involve potential merge conflicts, as two people might simultaneously make requests that change the same section of a post.

2 Reading

Before you work on this project you should read the chapter in the *Pro Git* book named *Git Branching*.

3 Details

I have created teams based on my own non-scientific observations of your writing, personalities, and the relationships I have observed. I will not comment on my criteria for doing this. Some of my decisions are based on a bit of social science, for what that is worth. The teams are as follows:

Team 1	**Team 2**	**Team 3**	**Team 4**
`audiencia-cereal`	`shadow12ac`	`johncgenere`	`f0cus10`
`DanieSegarra36`	`lashana29`	`Chocolate-Spaghet`	`FrancisXIrizarry`
`LiudmilaZyrianova239`	`anupamdas104`	`gutierrezjdr`	`Jimmyzs`
`yizongk`			

Suppose the members of the team are named x, y, and z, to make this easy.

1. Each of x, y, and z will fork a copy of the other two weekly blog post repositories, then clone these forks onto their local machines. Use the exact methods I described for the first git exercise to do this. Work only on the posts for the first two weeks, i.e., not the Git Workflow post.

2. Pick one of the two repositories to work on first. It is a good idea to coordinate with your team members to reduce the merge conflicts. For example, x can pick y, y can pick z, and z can pick x. Then they rotate so that x picks z, z picks y, and y picks x.

3. Read the post first for spelling errors, then for grammar errors you can detect, then usage. Grammar errors are sometimes obvious: "*Who have argued so well.*" should be "*Who has argued so well.*" Usage is when the wrong words or expressions are used in a sentence, like "*She should have written on the sorting algorithm, not on the searching algorithm.*" Here "*on*" should really be "*about*". If you think a sentence can be improved by a larger change, suggest it ***nicely***.

4. When you think your changes are good and complete, stage them (git add), commit them (with a good commit message), and then push them to your remote (git push origin). Go to `GitHub` and issue a pull request to the person whose post you have worked on.

5. When you, as the owner of your *GitHub* repository, receive a pull request from a teammate, say you are x and your teammate is y, you should pull the suggested changes down to your local machine. There are different ways to do this. Here is my suggestion.

   (a) On your repository page, click on the `Pull Requests` tab. You will see a list of pull requests, including the one from y. Click on the one from y.

   (b) If there is a merge conflict you will need to resolve it before proceeding with the remaining steps. You will see the message "`This branch has conflicts that must be resolved`" with a button labeled "`Resolve conflicts`" to its right. When you click on it, *GitHub* will change the window to one in which you can edit the text of the file and resolve the merge. Make changes so that the conflict does not exist and then click on the "`Commit Merge`" button. Continue to the next step.

   (c) Assuming that there are no merge conflicts, you should see a green `Merge Pull Request` button, to the right of which there is text reading "`view command line instructions`." The "`command line instructions`" text is clickable. Click on it and the page displays the command line instructions that are needed to pull down the code to be merged.

   (d) On your local machine, navigate to the directory containing your Git repository for the blog. You need to run two commands. The first creates a new branch in your repository, making it point to the current `master` branch, and makes it the current working branch (the one that `HEAD` points to.). You should name the new branch something like `y-suggested-changes`:

   ```
   git checkout -b y-suggestion-changes master
   ```

   The `master` at the end of this command tells Git to make the new branch point to the `master` branch.

   (e) For the next command, look at the URL for the name of y's repository from which the pull request comes. It will be something like

   ```
   https://github.com/y/x-weekly.git
   ```

   The command to run next pulls down the code to be merged into your new branch:

   ```
   git pull https://github.com/y/x-weekly.git master
   ```

   The `master` at the end of this command tells Git that the branch from which to pull the code is the `master` branch of y's repository. Unless the Git commands listed in the box state differently, you should use "master" in this command.

   (f) At this point you are ready to examine the changes proposed by y and make whatever changes you want to accept. You might want to open a discussion with y at this point. When you are satisfied, stage the file and commit them.

6. Before you push the changes up to your upstream repository, you will take the opportunity to use a new part of the workflow. Rather than pushing the changes to the master branch on the remote, you can create a branch on the remote on *GitHub*, and push the changes into that. To create a new branch in your remote repository on GitHub, click on the `Code` tab, then click in the `Branch:` box and type a new branch name. Suppose on *GitHub* you named this branch `y-branch`.

7. Back on your local machine, enter the command

   ```
   git push origin HEAD:y-branch
   ```

   Git will push the local repository into the new branch, `y-branch`, on the server.

8. At this point, you have a new version of you blog post in the y-branch on the server, based on the changes proposed by *y*. When *z* makes suggestions, you can repeat steps 1 through 6 above for *z*'s suggestions, working on your local machine. Then, you can merge all changes made by *y* and *z* into a single branch on the server, calling it something like revision. In order for the changes to be displayed on your GitHubPages webpage, these will have to be merged into the master branch on the server. You can do this as the final step.

No matter what you do when making suggestions, be thoughtful, considerate, and constructive. This is your chance to "start on the right foot" in the world of open source. Be a good citizen.

4 Due Date

I will be giving you another parallel assignment that requires reading and an activity. In the meanwhile start on this and have it finished by October 8, which is after we meet again. When you are finished, the person who owns the repository should open an issue on *GitHub* and send a comment to me that it is complete. If you need help figuring out how to do this, post a question to *Piazza*.

