Exercise 6 - Collaboration and conflict with GitHub

###Work with a partner to complete the tasks below and submit your results via a pull request on GitHub by the beginning of tutorial on Friday.

##Activity

1. Pick a partner and decide who is the repo **owner** and who is the repo **collaborator**.
2. **Owner** should `fork` the TA's repo on GitHub. This creates a version of the TA's repo in the owner's GitHub account.
3. **Owner** needs to invite the collaborator to work on the forked repo. This can be accomplished by clicking your small user icon in the upper right of your GitHub landing page and selecting "Your profile". Now click "Repositories" in the top middle of your profile, and click the name of the forked repo. Now select "Settings" in the top middle of the repo page and "Collaborators" on the left. Enter your collaborator's GitHub username or email address in the box and click "Add collaborator"
4. **Collaborator** check your email and accept the invite to collaborate. This WILL NOT create a repo in your GitHub account, but instead gives you access to the repo on the owner's GitHub account. You are now both allowed to `pull` and `push` content to this repo.
5. **Owner** and **Collaborator** now need to clone the forked repo (not the TA's repo!) to their computers in order to create a local Git repo linked to the GitHub repo. Use the command `git clone <repo url copied from GitHub>` to do this.
6. **Collaborator** should use `nano` to create a text file that includes their three favorite movies as the first three lines of the text file. Save this with an informative file name and commit a version of your Git repo. In order to update the GitHub repo the collaborator must now push their latest commit to GitHub. Use the command `git push origin master` to accomplish this. Git may ask you for your GitHub username and password at this point.
7. **Owner** should make sure their local repository is in sync with the GitHub repo by pulling it's contents to their local Git repo. Use `git pull origin master` to accomplish this. Now, add your favorite three movies to the fourth through sixth lines of the text file created by your collaborator. Commit your changes to your local Git repo and push those changes to GitHub.
8. **Collaborator** synchronize your Git and GitHub repos by pulling the current version down from GitHub.
9. **Owner** and **Collaborator** should now INDIVIDUALLY rank these movie titles to reflect their preferences. Put your favorite movie on the first line and second favorite on the second line, etc. DO NOT TALK TO EACH OTHER DURING THIS PROCESS!!!
10. **Owner** commit your changes to your local git repo and then push your changes to GitHub.
11. **Collaborator** after the repo owner has pushed their commits to Github, commit your changes to your local git repo and then push your changes to GitHub.
12. **What happened????** Because the two of you were working in parallel and editted the same lines of the file, the GitHub repo got out of sync with the collaborator's local repo and conflicts arose. These conflicts must be resolved before collaborator's changes can be pushed to GitHub. This is accomplished by pulling the GitHub repo contents and manually resolving conflicts identified by Git. If you didn't get any conflicts when the Collaborator attempted to push their repo to GitHub, why not? Repeat steps 9-11 in a manner that creates a conflict.

13. **Collaborator** pull the current version of the GitHub repo. You should see some messages indicating that there are conflicts that must be addressed.
14. **Collaborator** show the conflict messages to the repo owner and open the file with conflicts in `nano` and notice the markup inserted by Git that highlights the conflicts. Work with the repo owner to compromise on a ranking for the six movies. Commit your changes and push those changes to GitHub.

##Assignment Armed with your new GitHub collaboration skills, work with your partner to develop a single shell script that accomplishes the three tasks below. These tasks will require the file "wages.csv", which you should have in your local directory since you cloned the repo you forked from the TA.

1. Write a file containing the unique gender-yearsExperience combinations contained in the file "wages.csv". The file you create should contain gender in the first column and yearsExperience in a second column with a space separating the two columns. The rows should be sorted first by gender and then by yearsExperience, but remember to keep the pairings in a given row intact. Don't worry about column names in the output file.
2. Return the following information to `stdout` when the shell script is executed: 1) the gender, yearsExperience, and wage for the highest earner, 2) the gender, yearsExperience, and wage for the lowest earner, and 3) the number of females in the top ten earners in this data set. Be sure to indicate, which output is which when returning them to `stdout`.
3. Return one more piece of information to `stdout`: the effect of graduating college (12 vs. 16 years of school) on the minimum wage for earners in this dataset. Two hints: 1) you can assign the output of a pipeline to a variable with this code `variable_name=$(code)`, where variable_name can be any name of your choosing and "code" represents a Linux pipeline, and 2) you can assign numeric values to variables and then us the command `bc` to do simple arithmetic. If you've defined two shell variables (`val1` and `val2`) that are decimal values you can subtract them with the following code: `echo "$val1 - $val2" | bc`.

The **owner** should start by working on task #1 and **collaborator** should start by working on task #2. Do this in parrallel, not sequentially. Work together on task #3, and don't forget to check and edit each other's code from tasks #1 and #2. Remember to frequently `add-commit` locally and `push-pull` to GitHub to avoid conflicts. Also, remember you don't have to be in the same place at the same time to work on this collaboratively thanks to GitHub!!!

##Turning in your assignment via GitHub

Once you have committed all changes to your local Git repos and pushed all of those commits to the forked repo on GitHub, you can "turn in" your assignment using a `pull request`. This can be done from the GitHub repo website. When viewing the forked repo, select "Pull requests" in the upper middle of the screen, then click the green "New pull request" button in the upper right. You'll then see a screen with a history of commits for you and your collaborator, select the green "Create pull request button". In the text box next to your user icon near the top of the page, remove whatever text is there and add "owner's last name - collaborator's last name submission", but obviously substitute your last names. If I and Elizabeth Brooks worked on the project together the text would read "jones-brooks submission". Then click the green "Create pull request" button. **Only one of you will need to create a pull request.**