Git

Tips for instructors

- **Students cannot use collections and should only use the regular array.**
- **Students should use Git and GitHub in all exercises, regularly commit changes, and return to previous commits if needed.**

Question 1: Git Basics

1. Create a folder called learn_git.
2. go into the learn_git folder.
3. Create a file called third.txt.
4. Initialize an empty git repository.
5. Add third.txt to the staging area.
6. Commit with the message "adding third.txt".
7. Check out your commit with the git log.
8. Create another file called fourth.txt.
9. Add fourth.txt to the staging area.
10. Commit with the message "adding fourth.txt".
11. Remove the third.txt file.
12. Add this change to the staging area.
13. Commit with the message "removing third.txt".
14. Check out your commits using the git log.
15. Write the command to list all of the global configurations for git on your machine.

Question 2: GitHub

Create a remote repository and push your code from the local repo to the remote.

Question 3: Git Branching

Part 1: merge no-conflict

1. Use the **git branch** to see your branches.
2. Use the **branch** command to create a new branch.
3. Use the **switch** command to switch to it.
4. Make a couple of commits in the branch – perhaps adding a new file and/or editing existing ones.
5. Use the **log** command to see the latest commits. The two you just made should be at the top of the list.
6. Use the **switch** command to switch back to the main branch. Rerun the git log. Notice your commits don’t show up now. Check the files also – they should have their original contents
7. Use the switch command to switch back to your branch. Use the **gitk --all** or **git log --graph --all** to take a look at the commit-graph; notice it’s linear
8. Now switch to the main branch again. Use the merge command to merge your branch into it. Look for information about it having been a fast-forward merge. Look at the git log, and see that there is no merge commit. Take a look in gitk and see how the DAG(Directed acyclic graph) is linear.
9. Remove the branch after merging.

10. Use the **branch** command to create a new branch. Make a couple more commits.
11. **Switch** back to the main. Make a commit there, which should edit a different file from the ones you touched in your branch – to be sure there is no conflict.
12. Now **merge** your branch again. (Aside: you don't need to do anything to inform Git that you only want to merge things added since your previous merge. Due to the way Git works, that kind of issue simply does not come up, unlike in early versions of Subversion.)
13. Look at the git log. Notice that there is a merge commit. Also, look in gitk. Notice the DAG now shows how things forked, and then were joined up again by a merge commit.
14. Remove the branch after merging.

Part 2: merge with conflict

1. Use the **git branch** to see your branches.
2. Use the **branch** command to create a new branch.
3. Use the **switch** command to switch to it. Make a couple of commits.
4. Return to your main branch. Make a commit there that changes the exact same line, or lines, as commits in your branch did.
5. Now try to merge your branch. You should get a conflict.

6. Open the file(s) that is in conflict. Search for the conflict marker. Edit the file to remove the conflict markers and resolve the conflict.
7. Now try to commit. Notice that Git will not allow you to do this when you still have potentially unresolved conflicts. Look at the output of status too.
8. Use the add command to add the files that you have resolved conflicts in the staging area. Then use commit to commit the merge commit.
9. Take a look at the git log and gitk, and make sure things are as you expected.

Question 5: Search about how to pull a request in GitHub.

Question 6: Git stash

1. Initiate a local repository. make a couple of commits. change some files but don't commit these files.
2. Use git stash
3. Use git status
4. Then use git stash pop
