Name: John McCormack (CM#2329)

Partner: Richard Shomer

CSSE 376 - Lab 2:

1. Yes, I have worked with SVN mostly, and a little with Git before.

2. Yes, I have worked a limited amount with "Windows cmd."

3. The "Add" command tells Git that it should include the local changes to a specific file in the next "Commit" command. Git requires that each individually changed file be "added" to the commit stage if the developer wants those changes to be officially committed.

4. The "Commit" command actually commits the "added" files as *officially made changes to be saved.* However this commit is only *local* and the online repository does not know of these changes yet until the next stage described in the following question.

5. The "Push" command takes any previous *local* commits, and pushes them to the online, master repository to make them *global* to the entire team working on the repository.

6. There are two people on my team. This means there are three total repository copies. One for each team member, as well as the master, online copy on Git Hub makes three copies.

7. There are 3 commits in the repository's history.

8. Richard Shomer, my team member, created the second commit.

9. The second commit altered the README file. Specifically it added the phrase "First Change" with a new line of equal signs above it.

10. There are two members on my team. This means there are 3 branches total in the Git Hub copy of the repository. Each team member's branch and the master branch make three.

11. There are zero files with a student's username in the master branch. There is one file in each of the two, separate team-members' branches.

12. The "git branch" command creates a new *local* branch for a user to work on, independent of changes made to other branches until they decide to merge it with other branches. It remains local until the user commits and pushes that branch to GitHub online.

13. The “git checkout” command changes the working branch (checks out) to the branch name specified as a parameter after “git checkout.” It essentially changes your “workspace” to a different branch so that you can work independently of the other branches.

14. There are two members on my team. There are three versions of the README file (two team members + master version).

15. There are two members on my team. We performed two Git merges. One merge was fast-forward (the original, first merge) and the second merge had to be done manually (because it conflicted with the first merge).

16. Three branches exist.

17. None of the student branches are at the same point as the master branch. This is because all the merges were performed “inside of” or “from” the master branch “to” the other student branches. This essentially means that the master branch was the only one “taking” the change differences from the various merges.