CS 470 Distributed Consistency in Git Lab

Names: Josh Derrow & Brennan Krutis

1. Everyone in your group should sign up for a Github account. Everyone should also have access to a Linux or Mac OS X computer with Git installed. Use your lab machine or `stu` if your personal laptop does not fulfill these requirements. If you have never used Git before, you should configure your name and email first:

```
git config --global user.name "<first-name> <last-name>"
git config --global user.email "<email-address>"
```

2. Choose one person in your group to be the **owner**. Write their name here: Brennan

3. Have the owner in your group create a new repository on Github named "`cs470-git-lab`". Take note of the repository URL; it should look something like the following (click the green "Code" button to open a popup with this information):

```
https://github.com/lam2mo/cs470-git-lab.git        (HTTPS URL)
git@github.com:lam2mo/cs470-git-lab.git            (SSH URL)
```

4. The repository owner should add all of your group members to the repository in the "Collaborators" page under the "Settings" tab. All of the group members should receive an invitation email and will need to confirm the invitation.

5. Everyone should **clone** the (currently-empty) origin repository to their local machine using "`git clone <repo-url>`". If you are doing this exercise on your own, you should create at least two local copies by specifying unique folder names after the repository URL.

> **NOTE**: If you have an SSH key set up already, use the SSH URL. If you do not, you will need to create a temporary personal access token (PAT) and use the HTTPS URL. Here is a link to a tutorial that describes how to create a PAT: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

6. The owner of the repository should create a "`README.txt`" file in the root folder of the repository with some text in it. This person should **add** the file to the repository, **commit** this change to their local copy of the repository, and then **push** the changes to the remote origin repository using commands similar to the following:

```
git add README.txt
git commit -m "Initial commit"
git push
```

7. Everyone should **pull** updates from the remote repository to their local copy using "`git pull`" and verify that their local copy now contains a copy of the README file.

WARNING: DO NOT PROCEED TO THE NEXT STEP UNTIL YOU ARE CERTAIN THAT EVERYONE HAS A CLEAN COPY OF THE REPOSITORY WITH **ONLY** THE README FILE

8. Everyone should create a file in the root folder of the repository with some text in it; use your name as the filename. Each person should add that file to the repository and commit the change to their local copy using "`git add`" and "`git commit`" commands similar to the ones listed above. **Include your name somewhere in the commit message.** At this point, your local repositories have diverged. Verify that everyone has a different list of files in their local repository.

9. To restore consistency, all of your group members (or if you are working alone, each of your local copies) should take turns executing **pull** and **push** commands. Some members should be prompted at some point to approve a merge message. *(Advanced Git users: do not rebase! That somewhat defeats the purpose of this exercise.)*

10. What is the minimum number of pull and push operations required to synchronize all copies of the repository and restore consistency, **assuming your group has exactly three members**? (Your group may have a different size, but answer here as if it had three.) Give your answer and a one- or two-sentence rationale below.

Assuming our group had three people, it would require 3 pull operations and 3 push operations (6 total) (one pull and push for each person). In order to synchronize the repository, each person would have to push their local changes (3 pushes), then once everybody has pushed, each person can pull the updated repository (3 pulls).

11. Describe the consistency model that Git enforces. Which specific consistency model(s) from our class discussion is it most similar to? Discuss this among your group and write a paragraph or two explaining your answer below.

Git enforces a client-centric consistency model with monotonic reading and writing, in which every client has a complete copy of the entire repository, including its full history. This decentralized architecture means that consistency is maintained from the perspective of the individual client rather than being enforced globally by a central entity. Each client can work independently, committing changes locally without needing to communicate with others until they choose to push or pull changes.

This client-centric model emphasizes eventual consistency, which means that repositories will become consistent over time as changes are shared and merged, but temporary divergence (merge conflicts) can still occur between clients. This eventual consistency is achieved through explicit actions such as pulling, merging, and pushing. Any conflicts that do arise are handled manually, ensuring that all changes are intentional and verified before pushing. Overall, Git provides a high degree of flexibility because developers can work offline, experiment freely in branches, and manage changes in a distributed way. While Git does not guarantee strong consistency in the traditional centralized sense, it offers a powerful and scalable model that prioritizes developer autonomy and robustness over strict real-time synchronization.

12. Copy/paste the output of "`git log --graph`" here (or include a screenshot of the commit log graph from a visual tool such as gitk or GitX) to demonstrate that you have completed the assignment. At this point, the output should be identical regardless of which team member runs the command, but you should verify this before submitting.

```
*   commit ea23d72f554bfb0fb26c889fdfa1b4df962506d3 (HEAD -> main, origin/main)
|\  Merge: 6b35938 cd2d45b
| | Author: Josh D <derrowjb@dukes.jmu.edu>
| | Date:   Fri Apr 4 15:35:59 2025 -0400
| |
| |     Merge remote-tracking branch 'refs/remotes/origin/main'
| |
| * commit cd2d45b3e152616cffab1b06b24044b593d3b7c5
| | Author: Brennan Krutis <krutisbs@dukes.jmu.edu>
| | Date:   Fri Apr 4 15:30:39 2025 -0400
| |
| |     Brennan commit
| |
* | commit 6b35938be9634831db885efde4587cea6a7ffb75
|/  Author: Josh D <derrowjb@dukes.jmu.edu>
|   Date:   Fri Apr 4 15:33:56 2025 -0400
|
|       Josh First Commit
|
* commit 34875881ab761b6cd341e7688d14c2440697c4b9
  Author: Brennan Krutis <krutisbs@dukes.jmu.edu>
  Date:   Fri Apr 4 15:27:52 2025 -0400

      Initial commit
```

13. Submit this document as a PDF on Canvas by the deadline. If you worked in a group, please submit ONLY ONE copy per group and make sure the list of names at the top is accurate.