CS 470 Distributed Consistency in Git Lab

Names: Carson Blythe   Jakob Lind   Alden Geipel

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: Jakob Lindo

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.

7: Person 1 Pushes
Person 2 Pulls
Person 2 Pushes
Person 3 Pulls
Person 3 Pushes (Full repository now uploaded)
Person 2 Pulls
Person 1 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.

Data-Centric:

- Sequential Consistency
  - Each read (pull) requires a corresponding prior write (push)
  - The total order of these events will not be violated by Git. Git does not let you modify the main branch until you are at the end of the main branch's line of events. Essentially, Git enforces the need for the working repository to be 'up to date' before changes can be committed (any non-initial write must be preceded by a read). The main Git repository will only see events as though they were linearizable (i.e. have a total order). The order in which each individual node sees events do not violate this total order due to reads being enforced in order to create a new write.

```
P0:     W(x)a
P1:             R(x)a    W(x)b
P2:                              R(x)b
```

- Causal Consistency
  - Implied by Sequential

Client-Centric:

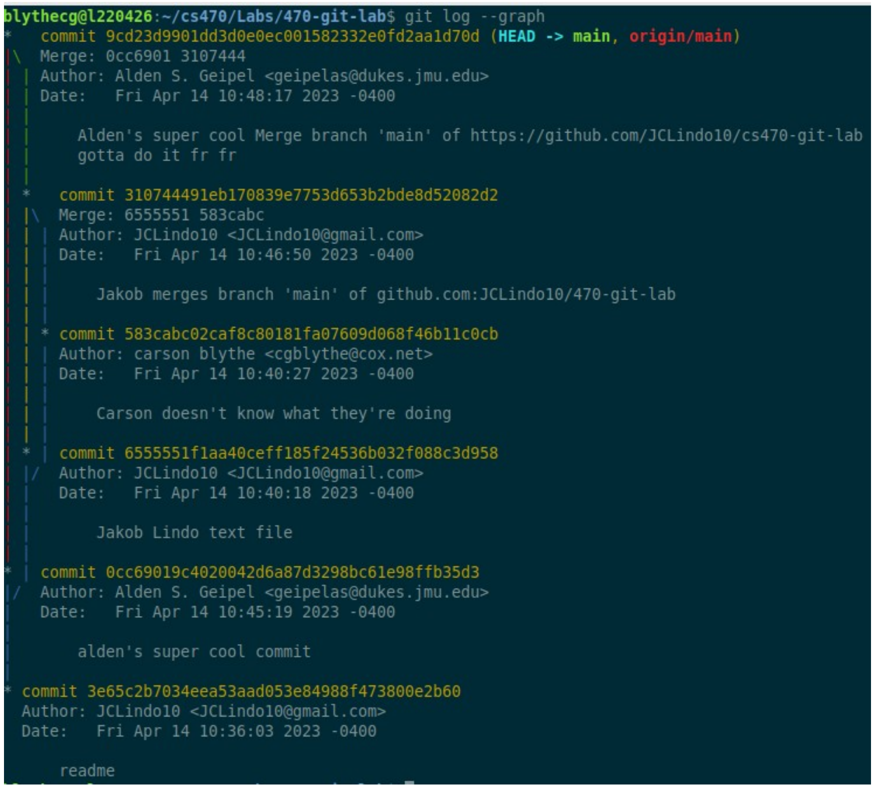
- Monotonic reads
  - Any read (pull) will see the most recent write (push). Git will say "Already up to date" rather than see an outdated version.
- Read-your-writes
  - Very similar to monotonic reads, if any person pushes to the repository, any pull will see the results of that push
- Writes-follow-reads
  - Git enforces a "read before write" policy, meaning any write must be on the current version. In short, any write to a Git repository must be read first, which will enforce the writes seeing the same or a newer version

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 9cd23d9901dd3d0e0ec001582332e0fd2aa1d70d (HEAD -> main, origin/main)
|\  Merge: 0cc6901 3107444
| | Author: Alden S. Geipel <geipelas@dukes.jmu.edu>
| | Date:   Fri Apr 14 10:48:17 2023 -0400
| |
| |     Alden's super cool Merge branch 'main' of https://github.com/JCLindo10/cs470-git-lab
| |     gotta do it fr fr
| |
| *   commit 310744491eb170839e7753d653b2bde8d52082d2
| |\  Merge: 6555551 583cabc
| | | Author: JCLindo10 <JCLindo10@gmail.com>
| | | Date:   Fri Apr 14 10:46:50 2023 -0400
| | |
| | |     Jakob merges branch 'main' of github.com:JCLindo10/470-git-lab
| | |
| | * commit 583cabc02caf8c80181fa07609d068f46b11c0cb
| | | Author: carson blythe <cgblythe@cox.net>
| | | Date:   Fri Apr 14 10:40:27 2023 -0400
| | |
| | |     Carson doesn't know what they're doing
| | |
| * | commit 6555551f1aa40ceff185f24536b032f088c3d958
| |/  Author: JCLindo10 <JCLindo10@gmail.com>
| |   Date:   Fri Apr 14 10:40:18 2023 -0400
| |
| |       Jakob Lindo text file
| |
* | commit 0cc69019c4020042d6a87d3298bc61e98ffb35d3
```

```
|/  Author: Alden S. Geipel <geipelas@dukes.jmu.edu>
|   Date:   Fri Apr 14 10:45:19 2023 -0400
|
|       alden's super cool commit
|
* commit 3e65c2b7034eea53aad053e84988f473800e2b60
  Author: JCLindo10 <JCLindo10@gmail.com>
  Date:   Fri Apr 14 10:36:03 2023 -0400

      readme
```



```
blythecg@l220426:~/cs470/Labs/470-git-lab$ git log --graph
*   commit 9cd23d9901dd3d0e0ec001582332e0fd2aa1d70d (HEAD -> main, origin/main)
|\  Merge: 0cc6901 3107444
| | Author: Alden S. Geipel <geipelas@dukes.jmu.edu>
| | Date:   Fri Apr 14 10:48:17 2023 -0400
| |
| |     Alden's super cool Merge branch 'main' of https://github.com/JCLindo10/cs470-git-lab
| |     gotta do it fr fr
| |
| *   commit 310744491eb170839e7753d653b2bde8d52082d2
| |\  Merge: 6555551 583cabc
| | | Author: JCLindo10 <JCLindo10@gmail.com>
| | | Date:   Fri Apr 14 10:46:50 2023 -0400
| | |
| | |     Jakob merges branch 'main' of github.com:JCLindo10/470-git-lab
| | |
| | * commit 583cabc02caf8c80181fa07609d068f46b11c0cb
| | | Author: carson blythe <cgblythe@cox.net>
| | | Date:   Fri Apr 14 10:40:27 2023 -0400
| | |
| | |     Carson doesn't know what they're doing
| | |
| * | commit 6555551f1aa40ceff185f24536b032f088c3d958
| |/  Author: JCLindo10 <JCLindo10@gmail.com>
| |   Date:   Fri Apr 14 10:40:18 2023 -0400
| |
| |       Jakob Lindo text file
| |
* | commit 0cc69019c4020042d6a87d3298bc61e98ffb35d3
|/  Author: Alden S. Geipel <geipelas@dukes.jmu.edu>
|   Date:   Fri Apr 14 10:45:19 2023 -0400
|
|       alden's super cool commit
|
* commit 3e65c2b7034eea53aad053e84988f473800e2b60
  Author: JCLindo10 <JCLindo10@gmail.com>
  Date:   Fri Apr 14 10:36:03 2023 -0400

      readme
```

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.