**Exerecise 3 Git-HOL**

**1. Introduction**

Git branching is a powerful feature that allows you to work on new features, bug fixes, or experiments in an isolated environment without affecting the stable codebase. Once your work is complete and tested, you can merge those changes back into the main branch. This approach prevents incomplete or broken code from being accidentally introduced into the project.

**Objectives**

* Create and switch between Git branches.
* Make commits on a feature branch.
* Merge a feature branch back into the main branch.
* Clean up unused branches.

**Prerequisites**

* **Git** must be installed and configured on your system.
* You should have an existing local Git repository. For this tutorial, we'll assume you have a repository called GitIgnoreDemo.

**Part 1: Branching**

**Step 1: Create a new branch**

Let's create a new branch called new-feature. This command creates the branch, but you will still be on the main branch.

git branch new-feature

**Step 2: List all branches**

Use this command to see all local and remote branches. The asterisk (\*) indicates the branch you are currently on.

git branch -a

**Step 3: Switch to the new branch**

Now, switch to the new-feature branch. Your working directory is now updated to reflect the state of this branch. Any new commits will be recorded here.

git switch new-feature

**Step 4: Create a new file**

To simulate development, create a new file named feature.html and add some content.

echo "<h1>New Feature Under Development</h1>" > feature.html

**Step 5: Stage the new file**

Add the new file to the staging area so Git knows to include it in the next commit.

git add feature.html

**Step 6: Commit the changes**

Commit the staged changes with a descriptive message.

git commit -m "feat: Add feature.html for a new development"

**Step 7: Check the branch status**

Use git status to confirm that your working directory is clean and all changes have been committed.

git status

**Part 2: Merging**

**Step 1: Switch back to the main branch**

Before merging your changes, you need to return to the main branch.

git switch main

**Step 2: Compare branches**

To see the differences between the main branch and the new-feature branch before merging, use git diff. You should see that feature.html exists in new-feature but not in main.

git diff new-feature

**Step 3: Merge the feature branch**

Now, integrate all the commits from the new-feature branch into the main branch.

git merge new-feature

**Step 4: View the commit history**

This command displays a concise, graphical view of your commit history, showing how the merge was performed.

git log --oneline --graph –decorate

**Step 5: Delete the merged branch**

Once the new-feature branch has been successfully merged, it's a good practice to delete it to keep your repository clean. The -d flag ensures that the branch is only deleted if it has been fully merged.

git branch -d new-feature

**Step 6: Confirm deletion**

Finally, run git branch again to confirm that the new-feature branch has been removed. You should only see the main branch listed.

git branch