**Git Branching and Merging Hands-On Assignment**

**PART 1: Branching**

**Step 1: Verify Master is in Clean State**

Before starting, I checked the status of the current branch (master) to ensure no pending changes:

git status

If the working directory is clean, it indicates I can proceed safely.

**Step 2: Create a Branch “GitWork” and Add hello.xml**

Now, I created a new branch named GitWork:

git branch GitWork

git checkout GitWork

Next, I created a file named hello.xml:

echo "<message>Hello, this is a Git test file.</message>" > hello.xml

**Step 3: Update Content in hello.xml and Check Status**

I made some changes to the file:

echo "<update>Updated content in GitWork branch</update>" >> hello.xml

Then I checked the status to verify the file is modified:

git status

**Step 4: Commit the Changes**

To reflect the changes in the branch, I staged and committed them:

git add hello.xml

git commit -m "Added and updated hello.xml in GitWork branch"

**Step 5: Switch Back to Master**

I returned to the master branch:

git checkout master

**Step 6: Add hello.xml to Master with Different Content**

echo "<message>Hello from the master branch</message>" > hello.xml

**Step 7: Commit the File to Master**

I committed the file in master:

git add hello.xml

git commit -m "Added hello.xml with different content in master"

**Step 8: Observe the Git Log**

I viewed a graphical log of all branches to visualize the commit history:

git log --oneline --graph --decorate --all

**Step 9: Compare Differences with Git Diff**

I used git diff to compare the contents of hello.xml in master and GitWork:

git diff GitWork

**Step 10: Visualize with P4Merge**

To better understand the differences, I used the P4Merge tool (after configuring it as the Git mergetool):

git mergetool

Make sure P4Merge is set as default in Git config:

git config --global merge.tool p4merge

git config --global mergetool.prompt false

**PART 2: Merging and Conflict Resolution**

**Step 11: Merge the Branch into Master**

While on master, I merged the GitWork branch:

git merge GitWork

Since both branches have conflicting versions of hello.xml, a merge conflict occurred.

**Step 12: Observe Git Conflict Markup**

Opening the hello.xml, I saw Git’s conflict markers:

<<<<<<< HEAD

<message>Hello from the master branch</message>

=======

<message>Hello, this is a Git test file.</message>

<update>Updated content in GitWork branch</update>

>>>>>>> GitWork

**Step 13: Resolve Conflict Using 3-Way Merge Tool**

I opened the P4Merge or any configured merge tool to resolve the conflict:

git mergetool

After resolving, I removed the conflict markers and finalized the content.

**Step 14: Commit the Merged Changes**

I staged and committed the merged file:

git add hello.xml

git commit -m "Merged GitWork into master with conflict resolution"

**Step 15: Update .gitignore to Ignore Backup Files**

Some merge tools create backup files like hello.xml.orig. I added such patterns to .gitignore:

echo "\*.orig" >> .gitignore

**Step 16: Commit .gitignore**

Then committed the ignore rule:

git add .gitignore

git commit -m "Ignore backup files created during merges"

**Step 17: List All Available Branches**

To confirm, I listed all branches:

git branch

**Step 18: Delete Merged Branch**

I deleted the now-merged branch:

git branch -d GitWork

**Step 19: View Final Git Log**

Finally, I verified the commit history with the graph view:

git log --oneline --graph --decorate