**4. Git-HOL**

**Step 1**: Verify if the master branch is in a clean state  
Command:  
 git checkout master  
 git status  
Explanation:  
Ensure there are no uncommitted changes in master before starting.

**Step 2**: Create a new branch named 'GitWork' and switch to it  
Command:  
 git checkout -b GitWork  
Explanation:  
This creates and switches to a new branch where changes will be made.

**Step 3**: Add a file 'hello.xml' in the GitWork branch  
Command:  
 echo '<message>Hello from GitWork</message>' > hello.xml  
 git status  
Explanation:  
Adds a new file to the branch.

**Step 4**: Commit the changes in GitWork  
Command:  
 git add hello.xml  
 git commit -m 'Add hello.xml in GitWork'  
Explanation:  
Saves changes to the branch history.

**Step 5**: Switch back to the master branch  
Command:  
 git checkout master  
Explanation:  
We will now create a different version of hello.xml in master.

**Step 6**: Add a different 'hello.xml' in master  
Command:  
 echo '<message>Hello from master</message>' > hello.xml  
 git add hello.xml  
 git commit -m 'Add hello.xml in master'  
Explanation:  
Creates a conflicting version of the same file.

**Step 7**: Observe the commit log  
Command:  
 git log --oneline --graph --decorate --all  
Explanation:  
Shows the commit history of all branches in a visual format.

**Step 8:** Compare master and GitWork branches  
Command:  
 git diff master GitWork  
Explanation:  
Displays differences between the two branches.

**Step 9:** Use P4Merge for visualization  
Command:  
 git mergetool --tool=p4merge  
Explanation:  
Opens P4Merge to see differences more clearly.

**Step 10**: Merge GitWork into master  
Command:  
 git merge GitWork  
Explanation:  
This will cause a merge conflict because hello.xml has different contents in both branches.

**Step 11:** Observe the merge conflict markers in hello.xml  
Explanation:  
The file will contain '<<<<<<<', '=======', and '>>>>>>>' markers showing the conflicting changes.

**Step 12**: Use 3-way merge tool to resolve conflict  
Command:  
 git mergetool  
Explanation:  
Opens the merge tool to manually choose changes from master, GitWork, or both.

**Step 13:** After resolving, mark the conflict as resolved and commit  
Command:  
 git add hello.xml  
 git commit -m 'Merge GitWork into master - conflict resolved'  
Explanation:  
Finalizes the merge.

**Step 14:** Add backup files to .gitignore  
Command:  
 echo '\*.orig' >> .gitignore  
 git add .gitignore  
 git commit -m 'Ignore merge backup files'  
Explanation:  
Prevents merge backup files from being tracked.

**Step 15:** List all branches  
Command:  
 git branch -a  
Explanation:  
Shows available branches.

**Step 16:** Delete GitWork branch after merge  
Command:  
 git branch -d GitWork  
Explanation:  
Removes the branch since it’s merged.

**Step 17:** Observe final log  
Command:  
 git log --oneline --graph --decorate  
Explanation:  
Shows the final repository state.