**GIT-HOL-4**

**CODE**

# Git Conflict Resolution Hands-on Lab

# Verify if master is in a clean state:

$git status

## Create a new branch “GitWork” and switch to it:

$git branch GitWork

$git checkout GitWork

## Add a new file hello.xml with some content:

$echo "<greeting>Hello from GitWork branch</greeting>" > hello.xml

##Check status to observe changes:

$git status

##Commit changes to the branch:

$git add hello.xml

$git commit -m "Add hello.xml in GitWork branch"

## Switch back to master:

$git checkout master

## Add a file hello.xml with different content than in branch:

$echo "<greeting>Hello from master branch</greeting>" > hello.xml

## Commit the changes to master:

$git add hello.xml

$git commit -m "Add hello.xml in master branch"

## Observe log graph for all branches:

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

## Check differences between branches with Git diff tool:

$git diff master GitWork

## Use P4Merge to visualize diff (ensure installed and configured):

$git difftool master GitWork

## Merge branch GitWork into master:

$git merge GitWork

##Git shows conflict markers in hello.xml. Use a 3-way merge tool to resolve:

$git mergetool

## After resolving, stage and commit the merge:

$git add hello.xml

$git commit -m "Resolve merge conflict in hello.xml"

## Check status and add backup files (e.g., hello.xml~) to .gitignore:

$git status

$echo "hello.xml~" >> .gitignore

$git add .gitignore

$git commit -m "Add backup files to .gitignore"

##List all branches:

$git branch

## Delete the merged branch:

$git branch -d GitWork

## Observe commit log graph:

$git log --oneline --graph --decorate

**OUTPUT**









