## Lab 1 – Git Configuration & Notepad++ Integration

This lab demonstrates how to set up Git configuration, integrate Notepad++ as the default editor, and add a file to a source code repository.

1. Check Git installation:  
    git --version
2. Configure user details:  
    git config --global user.name "Your Name"  
    git config --global user.email "your.email@example.com"
3. Verify configuration:  
    git config --global --list
4. Add Notepad++ path to Environment Variables if not recognized.
5. Set Notepad++ as default editor:  
    git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession"
6. Create new project folder and initialize Git:  
    mkdir GitDemo && cd GitDemo && git init
7. Create a file and verify content:  
    echo "This is my first Git file" > demo.txt
8. Add and commit file:  
    git add demo.txt  
    git commit -m "First commit: Added demo.txt"
9. Link with GitLab remote:  
    git remote add origin <repo-URL>
10. Push to remote:  
     git push -u origin main

## Lab 2 – Using .gitignore to Ignore Unwanted Files

This lab demonstrates how to use the .gitignore file to prevent tracking of certain files and folders in a Git repository.

1. Navigate to repository:  
    cd path/to/repo
2. Create unwanted files:  
    echo "Log file" > sample.log  
    mkdir log && echo "Error log" > log/error.log
3. Create/Edit .gitignore file with the following lines:  
    \*.log  
    log/
4. Check Git status to ensure files are ignored:  
    git status
5. If files were tracked before, remove from cache:  
    git rm --cached sample.log  
    git rm -r --cached log
6. Add and commit .gitignore:  
    git add .gitignore  
    git commit -m "Added .gitignore to ignore log files and log folder"
7. Push changes to remote:  
    git push origin main

## Lab 3 – Branching and Merging

This lab explains how to create a branch, make changes, and merge it into master (or trunk) in Git, as well as creating merge requests in GitLab.

1. Create new branch:  
    git branch GitNewBranch
2. List branches:  
    git branch -a
3. Switch to new branch:  
    git checkout GitNewBranch
4. Add files and commit:  
    echo "Branch file" > branchfile.txt  
    git add branchfile.txt  
    git commit -m "Added file in branch"
5. Switch to master:  
    git checkout master
6. Compare branches:  
    git diff master GitNewBranch
7. Merge branch to master:  
    git merge GitNewBranch
8. Delete branch:  
    git branch -d GitNewBranch
9. Push changes:  
    git push origin master

## Lab 4 – Conflict Resolution During Merge

This lab explains how to handle merge conflicts when changes in master conflict with changes in a branch.

1. Ensure master is clean:  
    git status
2. Create branch and file:  
    git checkout -b GitWork  
    echo "Branch content" > hello.xml  
    git add hello.xml  
    git commit -m "Branch version of hello.xml"
3. Switch to master and create different version:  
    git checkout master  
    echo "Master content" > hello.xml  
    git add hello.xml  
    git commit -m "Master version of hello.xml"
4. Merge branch:  
    git merge GitWork
5. Resolve conflict manually in hello.xml and mark as resolved:  
    git add hello.xml
6. Commit merge:  
    git commit -m "Resolved merge conflict"
7. Add backup file to .gitignore:  
    echo "\*.bak" >> .gitignore  
    git add .gitignore  
    git commit -m "Ignore backup files"
8. Delete branch:  
    git branch -d GitWork

## Lab 5 – Clean Up and Push to Remote

This lab explains how to verify a clean working state, pull changes, and push local changes to remote.

1. Check master state:  
    git status
2. List branches:  
    git branch -a
3. Pull latest from remote:  
    git pull origin master
4. Push pending changes:  
    git push origin master