Anusara Punchihewa
Department of Computer Science
University of Ruhuna

1. Make a folder as "**TestGIT**" in your Desktop and open the **gitbash** terminal.

2. Configure the local git repository.
   Global Configuration (Set your username and email for all repositories on your machine).

```
$ git config --global user.name <your_username>
$ git config --global user.email <your_email>

Ex:   $ git config --global user.name "Anusara_Punchihewa"
      $ git config --global user.email "anusara@email.com"
```

3. Set the default branch name as "main".

```
$ git config --global init.defaultBranch main
```

4. View current configuration details.

```
$ git config --global --list
```

5. Using the git bash application, create a git repository.

```
$ git init          // This command use to create a new repository (locally) in your current location
```

6. View current status of the working directory.

```
$ git status
```

7. Create a small java file(s).

```
$ touch Test.java          //You can create a single file or multiple files using touch command
```

8. Open the created file and define the class declaration.

```
$notepad Test.java              //Then include the class declaration and save it.
```

9. View current status and add modified content into the staging area (index).

```
$ git status                    //Check current status

$ git add .          OR         //stage all un-staged or modified files into index.
$ git add Test.java  OR         //stage single file.
$ git add Test.java Abc.java    //stage multiple files
```

10. Unstaged files in the index.

```
$ git restore --staged          //unstage all staged files upto last commit
$ git restore --staged Test.java //unstage selected file(s).
```

11. View current status and commit your changes into local git repository.

```
$ git status
$ git commit –m <commit message>

Ex: $ git commit –m "Test.java is created"
```

12. Perform several modifications into your project and commit the local git.

13. View the commit history.

```
$ git log
```

Anusara Punchihewa
Department of Computer Science
University of Ruhuna

14. View commit history in briefly.
    ```
    $ git log --oneline
    ```
15. Do some modifications into your working file(s) and add those changes to the previous commit.
    i. First make some changes to your project file(s).
    ii. Add those changes to index.
    iii. The commit those changes into previous commit
    ```
    $ git commit --amend
    ```
    //In here you can able to change the previous commit message also
    iv. The check the git log again to verify those changes are success.

16. Remove the last commit.
    ```
    $ git reset --soft HEAD~1
    ```
    //It will remove the last commit but the changes of the commit still remain
    ```
    $ git reset --soft HEAD~2
    ```
    //It will remove the 2nd previous commit to current commit

    Important: These commands can't be rollback.

17. Remove the last commit permanently.
    ```
    $ git reset --hard HEAD~1
    ```
    //It will remove the last commit and their changes. It will update to the current state as the one before commit.

    Important: These commands can't be rollback.

18. Perform following tasks.
    i. Compile Test.java.
    ii. Create following files.
    ```
    $ touch F1.txt F2.pdf
    ```
    iii. Check current status.
    iv. Create following folders.
    ```
    $ mkdir bin lib
    ```
    v. Check current status.
    vi. Create following files inside the bin folder.
    ```
    $ touch ./bin/F3.txt
    $ touch ./bin/F4.txt
    ```
    vii. Check current status.

19. Create a gitignore file (.gitignore) and check the status.
    ```
    $ touch .gitignore
    $ git status
    ```

20. Ignore all unnecessary files in the working directory.
    (Open the .gitignore file and type as given below and save it)
    ```
    #ignore F1.txt file
    ```
    //You can add (single line) comment into the .gitignore file using "#"
    ```
    F1.txt
    #ignore all .txt files
    *.txt
    #ignore all .class files
    *.class
    ```

21. Ignore folders and its content you have created above.
    ```
    bin/
    ```
    //ignore bin directory and its content
    ```
    lib/
    ```
    // ignore lib directory and its content

Anusara Punchihewa
Department of Computer Science
University of Ruhuna

22. Remove "F2.pdf" from git tracking.
    `$ git rm --cached F2.txt` //It will remove from git tracking in future commits but still in working area.
23. View all the tracked content so far in your local git.
    `$ git ls-files`

24. Compare the differences between working **directory** and **index.**
    `$ git diff` //It shows changes between working area and the index, if not it shows nothing.

25. Compare the differences between **index** and the **latest commit** (HEAD).
    `$ git diff --staged`

26. Compare the differences between **Commits**.
    `$ git diff <commit_1_ID> <commit_2_ID>`

GIT Branching

27. Get help about git branching.
    `$ git checkout –h`

28. Check available branches in your working directory.
    `$ git branch` //It will show you all available branches and current active branch

29. Create a separate branch called "version1".
    `$ git branch version1` //Creating a branch called "version1".
    `$ git switch version1` OR //Change the current branch to "version1".
    `$ git checkout version1`

    `$ git checkout –b version1` //Creating a branch called "version1" switched to it.

30. Update the branch "version1" by adding some content.

31. Go to the main branch and merge "version1" into it.
    `$ git checkout main`
    `$ git merge version1`

32. Delete the branch "version1".
    `$ git branch –d version1` //Delete the branch if it has been merged.
    `$ git branch –D version1` //Forcefully delete the branch, even if it has unmerged.

33. Now create another branch called "V2".
    `$ git checkout –b V2`

34. Modify the branch content.

35. Then change the branch to main and again modify its content.

36. Try to merge V2 into the main and see the result.

37. Open the Visual Code Studio and try to solve merge conflicts.

Anusara Punchihewa
Department of Computer Science
University of Ruhuna

Connecting to the Remote Repository

38. Log into your github account and make a new repository.

39. Make a connection to your local repository using https link.
    ```
    $ git remote add origin http://github.com/Anusara/Test1.git
    ```
    (Those command also available in you github account repository)

40. Push your local project to your github repository.
    ```
    $ git push -all
    ```

41. You can specifically select a branch and push it.
    ```
    $ git push –u origin main
    $ git push –u origin V1
    ```

42. Pull or clone the content to your local git from the remote git.
    ```
    $ git pull origin main
    ```

43. Finally, you can delete the local repository if you no longer need it.
    ```
    $ rm –rf .git
    ```