1- Initialize a new repository. Add two files in your working directory.

```
git init
```

2- View the status of the working directory and the staging area.

```
git status
```

3- Stage both files.

```
git add .
```

4- View the changes in the staging area.

```
git status
```

5- Create a commit.

```
git commit -m “first commit“
```

6- View the list of commits.

```
git log
```

7- View the content of the last commit.

```
git log -1
```

8- Update one of the files. View the changes in the working directory.

```
git status
```

9- Stage the changes.

```
git add .
```

10- Make another commit. Now you should have two commits in your repo.

```
git commit -m “second commit“
```

11- Show the changes in the last 2 commits.

```
git diff HEAD
```

12- Show all commits made by yourself. Use the one-liner option.

```
git log --author=khaki_mim
```

13- Show all commits with GUI in their message.

```
git gui
```

14- Show all commits with changes to file1.txt. Include the number of lines added/ removed.

```
git diff –numstat
```

15- Compare the last two commits.

git diff HEAD^ HEAD

16- Check out the commit before the last commit. Note the detached HEAD in the terminal. Check out the master branch.

git checkout -b feauter1

17- Show the author of every line in file1.txt.

git log --author="<khaki_mim>" --pretty=tformat: --numstat

18- Create a tag (v1.0) for the last commit. Show the history using the one-liner option and note the tag you just created.

git tag -a v1.0

19- Delete the tag.

git tag -d v1.0

20- Create a new branch called feature/login. Switch to the new branch.

git checkout -b feature/login

21- Show all the branches.

git branch -r

22- Update file1.txt in the current branch (feature/login) and make a new commit.

git add .

git commit -m "third commit"

23- Show the commits across all branches.

git log

24- Switch back to the master branch. Show the commits in the feature branch that don't exist on master.

git switch master

git log

25- View the differences between master and feature/login.

git diff master.. feature/login

26- Merge feature/login into master.

git merge feature/login

27- View the merged and unmerged branches.

git status

28- Delete the feature branch.

git branch -d feature/login

29- Create a new branch called feature/logout. On this branch, write blue to file1.txt and make a commit. Switch back to master, write green to file1.txt and make another commit. Try to merge your feature branch into master. You'll see a conflict. Resolve the conflict by accepting both changes. When you're done merging, delete the new branch.

git branch feature/logout

git switch feature/logout

Touch file1.txt

Nano

git add .

git commit -m "fourth commit"

git switch master

Nano

git add .

git merge feature/logout

accepting both changes conflict in vs code

git add .

git branch -d feature/logout

30- Create a new branch called bugfix/login. On this branch, write orange to file1.txt and make a commit. Switch back to master, write green to file2.txt and make another commit. View a graph of your branches. You'll see divergence. Rebase the new branch on top of master. View the graph of branches again. Note that the divergence is gone. Do a fast-forward merge to bring the changes in the bugfix branch into master.

git checkout -b bugfix/login

git add .

git commit -m "fifth commit"

git switch master

git add .

git commit -m "six commit"

git log –all –decorate –online – graph

git rebase master

git merge - - f master