1 Based on what you have learnt in the class, do the following steps:

a. Create a new folder
b. Put the following files in the folder

- Code.txt
- Log.txt
- Output.txt

c. Stage the Code.txt and Output.txt files
d. Commit them
e. And finally push them to GitHub

Ans : IN GIT BASH Terminal-

a. mkdir Git-Project
b. touch Code.txt Log.txt Output.txt
c. git add Code.txt Output.txt
d. git commit -m “added Code.txt and Output.txt files”
e. git add remote origin “https://remote-repo-url/”
f. git push origin main

2. Tasks to Be Performed:

1. Create a Git working directory with feature1.txt and feature2.txt in the master branch
2. Create 3 branches develop, feature1 and feature2
3. In develop branch create develop.txt, do not stage or commit it
4. Stash this file and check out to feature1 branch
5. Create new.txt file in feature1 branch, stage and commit this file
6. Checkout to develop, unstash this file and commit
7. Please submit all the Git commands used to do the above steps

Ans :

1. git init
2. touch feature1.txt feature2.txt
3. git branch -c develop - git branch -c feature1 - git branch -c feature2
4. git checkout develop
5. touch develop.txt
6. git add develop.txt – git stash
7. git checkout feature1
8. touch new.txt – git add new.txt – git commit -m “added new.txt file to feature1”

9. git checkout develop
10. git stash apply
11. git commit -m “adding unstashed file new.txt from feature1 branch to develop branch”

3. Tasks to Be Performed:

1. Create a Git working directory, with the following branches:

    - Develop
    - F1
    - f2

2. In the master branch, commit main.txt file
3. Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively
4. Push all these branches to GitHub
5. On local delete f2 branch
6. Delete the same branch on GitHub as well.

Ans :

1. git init – git branch -c Develop – git branch -c F1 – git branch -c f2
2. touch main.txt – git add main.txt – git commit -m “added main.txt file to master branch” (as we have not created branch using git checkout - we will still be in master branch even after creating branches)
3. git switch develop – touch develop.txt – git add . – git commit -m “added develop.txt”
4. git switch F1 – touch f1.txt – git add . – git commit -m “added f1.txt”
5. git switch f2 – touch f2.txt – git add . – git commit -m “added f2.txt”
6. git remote add origin https://remote repo url
7. git push --all origin

4. Tasks to Be Performed:

1. Put master.txt on master branch, stage and commit
2. Create 3 branches: public 1, public 2 and private
3. Put public1.txt on public 1 branch, stage and commit
4. Merge public 1 on master branch
5. Merge public 2 on master branch
6. Edit master.txt on private branch, stage and commit
7. Now update branch public 1 and public 2 with new master code in private
8. Also update new master code on master
9. Finally update all the code on the private branch

Ans :

1. git init
2. touch master.txt – git add . – git commit -m “added master.txt”
3. git branch -c public1 – git branch -c public2 – git branch -c private
4. git checkout public1 – touch public1.txt – git add . – git commit -m “adding public1.txt”
5. git checkout master – git merge public1 – git merge public2
6. git checkout private – touch master.txt – git add . – git commit -m “added master.txt”
7. git switch public1 – master file already there with the text which is in private branch - added some text
8. git switch public2 – master file already present with the modifications in private branch and public1 branch – modified again in the public2 branch
9. git switch master - master file contains all the changes are done in the previous branches – added some more text in the branch
10. git switch private – all the modifications done on the previous branches are already present in the master file

5. Tasks to Be Performed:

1. Create a Git Flow workflow architecture on Git
2. Create all the required branches
3. starting from e feature branch, push the branch to the master, following the architecture
4. Push an urgent.txt on master using hotfix

Ans :

GIT Workflow :

- Main branch for application code
- feature branch for developing features for the application
- admin branch for developing admin panel of the application
- Following with a PR request to merge the branches with the main application code

PW ASSIGNMENT - GIT

Creating Branches :

- git init
- touch application_code.txt (for the application code)
- git add .
- git commit -m “Base application code”
- git branch -c features
- touch feature1.txt
- git add .
- git commit -m “feature 1 of the application”
- git branch -c admin
- git add .
- git commit -m “admin panel of the application”

Pushing code to the main branch :

- git checkout main
- git merge feature1
- git merge admin

Pushing on main branch using Hot Fix branch :

- git pull origin main (if on remote repo - to get the latest main branch)
- git checkout -b HotFix
- touch urgent.txt
- git add .
- git commit -m “urgent fix to the application”
- git checkout main
- git merge hotfix
- git push origin main