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.**

Based on what i have learned in class, do the following steps:

1. **Create a new folder**:
   - Command: mkdir new folder
2. **Put the following files in the folder**:
   - Create files:
     - Command: touch Code.txt Log.txt Output.txt
3. **Stage the Code.txt and Output.txt files**:
   - Command: git add Code.txt Output.txt
4. **Commit them**:
   - Command: git commit -m "Added Code.txt and Output.txt"
5. **Push them to GitHub**:
   - Command:
     - git remote add origin <remote-repository-URL>
     - git push -u origin master

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.

Create a Git working directory with feature1.txt and feature2.txt in the master branch:

Create a Git working directory:

- Command:
  - mkdir project
  - cd project
  - git init

1. Create feature1.txt and feature2.txt:
   - Command:
     - touch feature1.txt feature2.txt
2. Create branches develop, feature1, and feature2:
   - Command:
     - git branch develop
     - git branch feature1
     - git branch feature2
3. In the develop branch, create develop.txt, do not stage or commit it:
   - Command:
     - git checkout develop
     - touch develop.txt

4.Stash this file and check out to feature1 branch:

- Command:
  - git stash
  - git checkout feature1

5.Create new.txt in feature1 branch, stage and commit this file:

- Command:
  - touch new.txt
  - git add new.txt
  - git commit -m "Added new.txt in feature1"

6. Checkout to develop, unstash this file and commit:
   - Command:
     - git checkout develop
     - git stash pop
     - git add develop.txt
     - git commit -m "Added develop.txt"
7. All the Git commands used to do the above steps:

   Command:
   - git init
   - git branch “branch name”
   - git checkout “branch name”
   - git stash
   - git add “file name”
   - git commit -m "file name"

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. put all these branches to Github

5. on local delete f2 branch

6. delete the same branch on Github as well.

Create a Git working directory with Develop, F1, and F2 branches:

1. **Create branches**:
    - Command:
        - git branch develop
        - git branch F1
        - git branch F2
2. **In the master branch, commit main.txt**:
    - Command:
        - touch main.txt
        - git add main.txt
        - git commit -m "Added main.txt"
3. **Put develop.txt in develop, f1.txt in F1, and f2.txt in F2**:
    - Command:
        - git checkout develop
        - touch develop.txt
        - git add develop.txt
        - git commit -m "Added develop.txt"
        - git checkout F1
        - touch f1.txt
        - git add f1.txt
        - git commit -m "Added f1.txt"
        - git checkout F2
        - touch f2.txt
        - git add f2.txt
        - git commit -m "Added f2.txt"
4. **Push all these branches to GitHub**:

- Command:
  - git push origin master
  - git push origin develop
  - git push origin F1
  - git push origin F2

5. **On local, delete f2 branch**:
   - Command:
     - git branch -d F2
6. **Delete the same branch on GitHub**:
   - Command:
     - git push origin --delete F2

4. Tasks to be performed:

1. put master.txt on master branch, stage and commit

2. create 3 branches: public 1, public 2, private

3. put public.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.

1. **Put master.txt on master branch, stage and commit**:
   - Command:
     - touch master.txt
     - git add master.txt
     - git commit -m "Added master.txt"

2. **Create 3 branches: public1, public2, and private**:
    - Command:
        - git branch public1
        - git branch public2
        - git branch private
3. **Put public1.txt on public1 branch, stage and commit**:
    - Command:
        - git checkout public1
        - touch public1.txt
        - git add public1.txt
        - git commit -m "Added public1.txt"
4. **Merge public1 with the master branch**:
    - Command:
        - git checkout master
        - git merge public1
5. **Merge public2 with the master branch**:
    - Command:
        - git checkout public2
        - touch public2.txt
        - git add public2.txt
        - git commit -m "Added public2.txt"
        - git checkout master
        - git merge public2
6. **Edit master.txt on the private branch, stage and commit**:
    - Command:
        - git checkout private
        - echo "Update private code" >> master.txt
        - git add master.txt
        - git commit -m "Updated master.txt on private"
7. **Update branch public1 and public2 with the new master code**:

- Command:
  - git checkout public1
  - git merge master
  - git checkout public2
  - git merge master

8. **Update the master branch code on the private branch**:
   - Command:
     - git checkout private
     - git merge master
9. **Finally, update all the code on the private branch**:
   - Command:
     - git checkout private
     - git merge public1
     - git merge public2

5. Tasks to be performed:

1. create a Git flow workflow architecture on Git

2. create all the required branches

3. starting from a feature branch, push the branch to the master, following the architecture

4. push an urgent.txt on master using hotfix

Create a Git Flow workflow architecture:

1. **Create all the required branches**:
   - Command:
     - git branch develop
     - git branch feature

        - git branch release
        - git branch hotfix

2. **Starting from the feature branch, push the branch to master, following the architecture**:
    - Command:
        - git checkout feature
        - git commit -m "Feature complete"
        - git checkout master
        - git merge feature

3. **Push an urgent.txt on the master branch using hotfix**:
    - Command:
        - git checkout hotfix
        - touch urgent.txt
        - git add urgent.txt
        - git commit -m "Urgent fix"
        - git checkout master
        - git merge hotfix