1. 1st Answer
    a. Create a new folder
    mkdir MyProjectFolder
    cd MyProjectFolder

    b. Put the following files in the folder
    touch Code.txt Log.txt Output.txt

    c. Stage the Code.txt and Output.txt files
    git init # Initialize a new Git repository
    git add Code.txt Output.txt # Stage only these two files

    d. Commit them
    git commit -m "Add Code.txt and Output.txt"

    e. Push them to GitHub
    git remote add origin https://github.com/your-username/MyProjectFolder.git
    git branch -M main # rename to main if needed
    git push -u origin main

2. 2nd Answer
    a. Create a Git working directory with feature1.txt and feature2.txt in the master branch
    mkdir GitWorkflow
    cd GitWorkflow
    git init

    touch feature1.txt feature2.txt
    git add feature1.txt feature2.txt
    git commit -m "Add feature1.txt and feature2.txt to master branch"

    b. Create 3 branches: develop, feature1, and feature2
    git branch develop
    git branch feature1
    git branch feature2

    c. In develop branch, create develop.txt, do not stage or commit it
    git checkout develop
    touch develop.txt
    # Do not add or commit

    d. Stash this file and check out to feature1 branch

    git stash push -m "Uncommitted develop.txt"

    git checkout feature1

e. Create new.txt file in feature1 branch, stage and commit it
   touch new.txt
   git add new.txt
   git commit -m "Add new.txt to feature1 branch"

f. Checkout to develop, unstash the file, and commit it
   git checkout develop
   git stash pop # This restores develop.txt
   git add develop.txt
   git commit -m "Add develop.txt to develop branch"

3. 3rd Answer
   a. Create a Git working directory with branches: develop, f1, f2
      mkdir GitProject
      cd GitProject
      git init

      # Create initial commit (required before creating branches)
      touch .gitkeep
      git add .gitkeep
      git commit -m "Initial commit"

      # Create the branches
      git branch develop
      git branch f1
      git branch f2

   b. In the master branch, commit main.txt
      touch main.txt
      git add main.txt
      git commit -m "Add main.txt in master branch"

   c. Add develop.txt to develop, f1.txt to f1, and f2.txt to f2
      # Develop branch
      git checkout develop
      touch develop.txt
      git add develop.txt
      git commit -m "Add develop.txt to develop branch"

      # F1 branch
      git checkout f1
      touch f1.txt
      git add f1.txt
      git commit -m "Add f1.txt to f1 branch"

      # F2 branch
      git checkout f2

touch f2.txt
git add f2.txt
git commit -m "Add f2.txt to f2 branch"

d. Push all branches to GitHub
git remote add origin https://github.com/your-username/GitProject.git

git push -u origin master
git push -u origin develop
git push -u origin f1
git push -u origin f2

e. Delete the f2 branch locally
git branch -d f2

f. Delete the f2 branch on GitHub

git push origin --delete f2

4. 4th Answer

mkdir ProjectRepo
cd ProjectRepo
git init

touch master.txt
git add master.txt
git commit -m "Add master.txt on master branch"

git branch public1
git branch public2
git branch private

git checkout public1
touch public1.txt
git add public1.txt
git commit -m "Add public1.txt on public1 branch"

git checkout master
git merge public1 -m "Merge public1 into master"
git merge public2 -m "Merge public2 into master"

git checkout private
echo "Private branch changes" >> master.txt
git add master.txt
git commit -m "Update master.txt in private branch"

git checkout master

```
git merge private -m "Merge updates from private into master"

git checkout public1
git merge master -m "Sync public1 with latest master"

git checkout public2
git merge master -m "Sync public2 with latest master"

git checkout private
git merge master -m "Update private with all changes from master"
```

5. 5th Answer
    a. Create a Git Flow Workflow Architecture

        Git Flow typically uses the following branches:

- master – production-ready code
- develop – integration branch for features
- feature/* – feature development branches
- release/* – pre-production branches
- hotfix/* – emergency fixes to production

    b. Create all the required branches

```
mkdir GitFlowProject
cd GitFlowProject
git init

# Add initial commit
touch README.md
git add README.md
git commit -m "Initial commit on master"

# Create develop branch from master
git branch develop
```

    c. Start from a feature branch and push to master following Git Flow

```
git checkout develop
git checkout -b feature/login-page

touch login.txt
git add login.txt
git commit -m "Add login page feature"

git checkout develop
git merge feature/login-page -m "Merge login feature into develop"
```

```
git checkout -b release/v1.0

touch changelog.txt
git add changelog.txt
git commit -m "Add changelog for v1.0"

git checkout master
git merge release/v1.0 -m "Release v1.0 to production"

git checkout develop
git merge release/v1.0 -m "Sync develop with v1.0 release"
```

d. Push an urgent.txt on master using hotfix

```
git checkout master
git checkout -b hotfix/fix-critical-issue

touch urgent.txt
echo "Urgent patch" > urgent.txt
git add urgent.txt
git commit -m "Hotfix: Add urgent.txt to master"

git checkout master
git merge hotfix/fix-critical-issue -m "Apply hotfix to master"

git checkout develop
git merge hotfix/fix-critical-issue -m "Apply hotfix to develop"
```

e. Push to GitHub

```
git remote add origin https://github.com/your-username/GitFlowProject.git
git push -u origin master
git push -u origin develop
git push -u origin feature/login-page
git push -u origin release/v1.0
git push -u origin hotfix/fix-critical-issue
```