Q1

1.Tasks to Be Performed:

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

a. Create a new folder

mkdir Myproject

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

cd MyProject

touch Code.txt Log.txt Output.txt

c. Stage the Code.txt and Output.txt files

git init

git add Code.txt Output.txt

d. Commit them

git commit -m "Added Code.txt and Output.txt files"

e.Push them to GitHub

- First, create a new repository on GitHub named "MyProject"
- Copy the remote URL and run:

  bash: git remote add origin https://github.com/AdnanHajwani12/MyProject.git

git branch -M main

git push -u origin main

Double-click (or enter) to edit

Q2

Tasks to Be Performed:

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

```
mkdir Git-Project

cd Git-Project

git init

touch feature1.txt feature2.txt

echo "This is feature 1" > feature1.txt

echo "This is feature 2" > feature2.txt

git add feature1.txt feature2.txt

git commit -m "Add feature1.txt and feature2.txt to master"
```

2. Create 3 branches develop, feature1 and feature2

```
git branch develop
git branch feature1
git branch feature2
```

3. In develop branch create develop.txt, do not stage or commit it

```
git checkout develop
touch develop.txt
echo "Development in progress" > develop.txt
```

4. Stash this file and check out to feature1 branch

```
git stash
git checkout feature1
```

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

```
touch new.txt
echo "This is new feature work" > new.txt
```

```
git add new.txt
git commit -m "Add new.txt to feature1 branch"
```

6. Checkout to develop, unstash this file and commit

```
git checkout develop
git stash pop
git add develop.txt
git commit -m "Add develop.txt after unstashing"
```

Q3

Tasks to Be Performed:

1. Create a Git working directory, with the following branches: Develop|F1|f2

```
mkdir Git-MultiBranch-Task
cd Git-MultiBranch-Task
git init
```

2. In the master branch, commit main.txt file

```
touch main.txt
echo "This is the main file" > main.txt
git add main.txt
git commit -m "Add main.txt to master"
```

3. Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively

```
git checkout -b develop
touch develop.txt
echo "This is the develop branch file" > develop.txt
git add develop.txt
git commit -m "Add develop.txt to develop branch"

git checkout -b f1
touch f1.txt
echo "This is F1 branch file" > f1.txt
git add f1.txt
```

```
git commit -m "Add f1.txt to f1 branch"

git checkout -b f2
touch f2.txt
echo "This is F2 branch file" > f2.txt
git add f2.txt
git commit -m "Add f2.txt to f2 branch"
```

4. Push all these branches to GitHub

```
git remote add origin https://github.com/YOUR_USERNAME/YOUR_REPO_NAME.git
git push -u origin master
git push -u origin develop
git push -u origin f1
git push -u origin f2
```

5. On local delete f2 branch

```
git branch -d f2
git push origin --delete f2
```

Q4

Tasks to Be Performed:

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

```
git checkout master

echo "This is master.txt" > master.txt

git add master.txt

git commit -m "Add master.txt to master"
```

2. Create 3 branches: public 1, public 2 and private

```
git branch public1
git branch public2
git branch private
```

3. Put public1.txt on public 1 branch, stage and commit

```
git checkout public1
echo "This is public1.txt" > public1.txt
git add public1.txt
git commit -m "Add public1.txt to public1"
```

4. Merge public 1 on master branch

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

5.Merge public 2 on master branch

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

6. Edit master.txt on private branch, stage and commit

```
git checkout private
echo "Updated by private branch" >> master.txt
git add master.txt
git commit -m "Edit master.txt on private branch"
```

7. Now update branch public 1 and public 2 with new master code in private

```
git checkout public1
git merge private -m "Update public1 with private changes"

git checkout public2
git merge private -m "Update public2 with private changes"
```

8. Also update new master code on master

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

9. Finally update all the code on the private branch

```
git checkout private
git merge master -m "Update private with latest master code"
```

Q5

Tasks to Be Performed:

1. Create a Git Flow workflow architecture on Git

Git Flow is a branching model for Git that helps manage features, releases, and hotfixes systematically. It consists of the following main branches:

```
- **`master`** – production-ready code
- **`develop`** – integration branch for features
- **`feature/*`** – for new features
- **`release/*`** – for preparing new production releases
- **`hotfix/*`** – for urgent fixes
```

2. Create all the required branches

```
git init git-flow-project
cd git-flow-project
```

- **Create main branches:**

```
git checkout -b master
git checkout -b develop
```

- **Push the branches to the remote:**

```
git remote add origin <your-repo-URL>
git push -u origin master
git push -u origin develop
```

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