1) Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

i) Create a new folder called “aec” and switch into the folder
mkdir aec
cd aec

ii) To initialize git repository
git init

iii) Create a new file called “hello.txt”
touch hello.txt

iv) Add the file to the staging area
git add .

v) Committing the file
git commit -m “initial commit: adds a hello file”

2) Creating and Managing Branches
   a) Create a new branch named "feature-branch." Switch to the "master" branch. Merge the "featurebranch" into "master." (Here the name can be either master or main)

i) Create a new folder called “aec” and switch into the folder
mkdir aec
cd aec

ii) To initialize git repository
git init

iii) Create a new file called “hello.txt”
touch hello.txt

iv) Add the file to the staging area
git add .

v) Committing the file
git commit -m “adds a hello file”

vii) Create and switch to the feature-branch
git checkout -b feature-branch

viii) Create a new file called “hello2.txt”
touch hello2.txt

ix) Add the file to the staging area
git add .

x) Committing the file
git commit -m “adds a hello2 file”

xi) Create and switch to the feature-branch
git checkout main

xii) Merge the branch to the main branch
git merge feature-branch

(Both hello1 and hello2 file should be present)
git log - to see the progress

b) Write the commands to stash your changes, switch branches, and then apply the stashed changes.

xiii) Create a new file
touch hello3.txt

ivx) Add the changes
git add .

xv) Stash the changes
git stash
(The file will go)

xvi) Checkout the feature branch
git stash pop

(The file should appear back in this branch)

3) Collaboration and Remote Repositories

a) Clone a remote Git repository to your local machine.

git clone https://github.com/dev-shetty/aec-github.git
cd aec-github

b) Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

git fetch
git checkout -b feature-branch
git rebase origin/master

c) Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.

touch hello.txt
git add .
git commit -m “adds hello file”
git checkout master
git rebase feature-branch

4) Git Tags and Releases Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository

i) Create a new folder called “aec” and switch into the folder
mkdir aec
cd aec

ii) To initialize git repository
git init

iii) Create a new file called “hello.txt”
touch hello.txt

vi) Add the file to the staging area
git add .

v) Committing the file
git commit -m “adds a hello file”

vi) Tag the commit
git tag v1.0

(To see the tag run **git log**, v1.0 will be present there)

5) Advanced Git Operations Write the command to cherry-pick a range of commits from "source-branch" to the current branch

i) Create a new folder called “aec” and switch into the folder
mkdir aec
cd aec

ii) To initialize git repository
git init

iii) Create a new file called “hello.txt”
touch hello.txt

iv) Add the file to the staging area
git add .

v) Committing the file
git commit -m “adds a hello file”

vi) Copy the commit hash from the git log
git log

```
deveesh@deveesh in repo: aec on master on (ap-south-1) on sosc@sahyadri.edu.in took 9ms
λ git log
commit 5fe82f368b32e0b691ae7fe626941182c034c354 (HEAD → master, tag: v1.0)
Author: Deveesh Shetty <deveeshshetty@gmail.com>
Date:   Wed Mar 13 17:20:14 2024 +0530
```

Here the commit hash is the full word from 5fe8….c354 (copy the entire thing it will be different in your commit)

vi) Cherry pick the commit
git cherry-pick commit-hash
(Replace the commit-hash with the copied commit hash, should get message like this)

```
┌─deveesh@deveesh in repo: aec on ⎇ master on ☁ (ap-south-1) on ☁   sosc@sahyadri.edu.in took 7ms
└─λ git cherry-pick 5fe82f368b32e0b691ae7fe626941182c034c354
On branch master
You are currently cherry-picking commit 5fe82f3.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'
```

6) Analysing and Changing Git History

i) Create a new folder called “aec” and switch into the folder
mkdir aec
cd aec

ii) To initialize git repository
git init

iii) Create a new file called “hello.txt”
touch hello.txt

iv) Add the file to the staging area
git add .

v) Committing the file
git commit -m “adds a hello file”

vi) Copy any of the commit hash by doing
git log

```
deveesh@deveesh in repo: aec on master on (ap-south-1) on  sosc@sahyadri.edu.in took 9ms
λ git log
commit 5fe82f368b32e0b691ae7fe626941182c034c354 (HEAD → master, tag: v1.0)
Author: Deveesh Shetty <deveeshshetty@gmail.com>
Date:   Wed Mar 13 17:20:14 2024 +0530
```

a) Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?
git show commit-hash

b) Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."
git log --author="JohnDoe" --since="2023-01-01" --until="2023-12-31"

c) Write the command to display the last five commits in the repository's history.
git log -n 5

d) Write the command to undo the changes introduced by the commit with the ID "abc123"

git revert commit-hash