**GIT ASSIGNMENT**

**Question 1**

Step 1: Create a new Git repository.

Step 2: Create a file and commit changes.

Step 3: View the commit history of your repository.

Step 4: Open the file you created earlier and make some changes to it.

Step 5: Check the file you modified is now marked as "modified" and unstaged.

Hint (git status)

Step 6: Stage the changes you made to the file and commit the changes to the repository.

Step 7: Clone the repository you have created in GitHub.

Step 8: Fetch the changes, navigate into the cloned repository using the command line, and use the command git fetch to fetch any changes that have been made to the original repository since you cloned it.

Step 9: Pull changes, merge the changes you just fetched into your local copy of the repository, and use the command git pull.

Solution of Question 1: -

Step 1: Create a new Git repository: -

• Open your terminal or command prompt.

. Set Your Username and email address as Git Configuration.

A black screen with white text

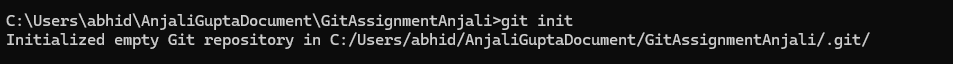
Description automatically generated

• Navigate to the directory where you want to create the repository.

A screen shot of a computer code

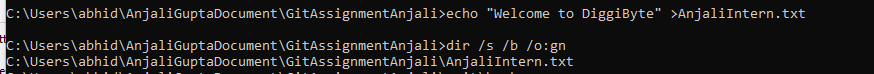
Description automatically generated

• Run the command git init. This initializes a new Git repository in the current directory.



Step 2: Create a file and commit changes.

• Create a new file in the repository directory by command prompt with some text message



• Run the following commands: git status

git add <filename>

git commit -m "Initial commit" // This commits the changes with a message

git status

A screen shot of a computer

Description automatically generated

Step 3: View the commit history of your repository.

• Run the command

git log.

Notes:- This will display a list of commits along with their details such as commit id hash, author, date, and commit message.

A computer screen with white text and numbers

Description automatically generated

Step 4: Open the file “AnjaliIntern.txt” and make some changes to it.

Before Changes:-

A screenshot of a computer

Description automatically generated

After Changes: -

A screenshot of a computer

Description automatically generated

Step 5: Check the file you modified is now marked as "modified" and unstaged.

• Run the command

git status.

Note :- This command shows the status of your working directory and staging area. It will display the modified file as unstaged.

A screen shot of a computer

Description automatically generated

Step 6: Stage the changes you made to the file and commit the changes to the repository.

• Run the following commands:

git add AnjaliIntern.txt

git commit -m "Modified file" // Commit the changes with an appropriate message.

A screen shot of a computer code

Description automatically generated

Step 7: Clone the repository you have created on GitHub.

• Go to GitHub and navigate to the repository you want to clone.

A screenshot of a computer

Description automatically generated

• Click on the "Code" button and copy the repository URL.

• In your terminal or command prompt, navigate to the directory where you want to clone the repository.

• Run the command git clone <repository URL>.

A black screen with colorful text

Description automatically generated

Step 8: Fetch the changes, navigate into the cloned repository using the command line, and use the command git fetch to fetch any changes that have been made to the original repository since you cloned it.

• Navigate into the cloned repository directory using the command line.

• Run the command git fetch origin. This will fetch any changes from the remote repository named "origin".

A screen shot of a computer

Description automatically generated

Step 9: Pull changes, merge the changes you just fetched into your local copy of the repository, and use the command git pull.

A screen shot of a computer

Description automatically generated

**A computer screen with white text

Description automatically generated**

**Question 2**

Step 1: Clone the repository you have created in GitHub.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Step 2: Create a new branch using the command.

Navigate into the cloned repository directory using the command line.

Run the following command to create a new branch:

Syntax :- git checkout -b new-branch

A screen shot of a computer code

Description automatically generated

Step 3: Switch to the new branch.

We can switch to it using the following command:

Syntax :- git checkout new-branch

A screen shot of a computer

Description automatically generated

Step 4: Make some changes to the code in your local copy of the repository.

Create new program and run it

A screenshot of a computer program

Description automatically generated

Step 5: Commit changes to the new branch.

Run below command for committing these changes.:-

Syntax:- git add .

git commit -m "Your commit message".

A screen shot of a computer screen

Description automatically generated

Step 6: Switch back to the original branch

We can run using the following command:

Syntax:- git checkout original-branch

A screen shot of a computer code

Description automatically generated

Step 7: Merge the new branch.

Once we're on the original branch, we can merge the changes from the new branch into it using the following command:

Syntax:- git merge new-branch

This command will merge the changes from the "new-branch" into the current branch

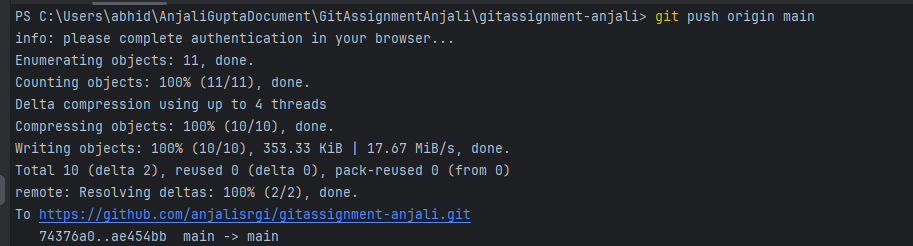
A black screen with blue and white text

Description automatically generated

Step 8: Push changes to the original branch

Finally, we can push the changes from the original branch to the remote repository using the following command:

Syntax :- git push origin original-branch

****

**Question 3**

Step 1: Create a feature branch.

Navigate to the directory of your Git repository.

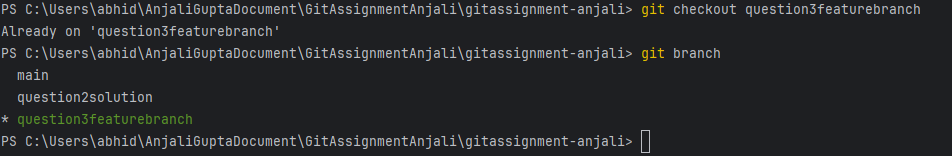
Run the following command to create a new feature branch:

Syntax : git checkout -b feature-branch



Step 2: Switch to the new branch.

Syntax : git checkout feature-branch



Step 3: open the file and make some changes to it.

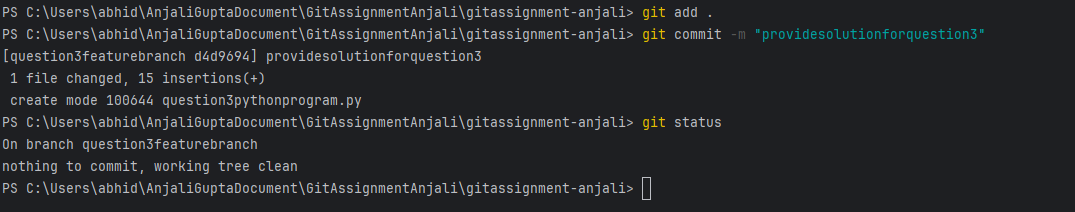
A screen shot of a computer program

Description automatically generated

Step 4: Add and commit the changes to the new branch.

Syntax : git add .

git commit -m "Your commit message" .



Step 5: Push the changes to the new feature branch.

Once i've committed my changes to the feature branch, i push them to the remote repository using the following command:

Syntax : git push origin feature-branch

A computer screen shot of a program

Description automatically generated

Step 6: Create a pull request.

Go to the GitHub repository in your web browser.

Switch to the "Pull Requests" tab And Click "New pull request" button.

A screenshot of a computer

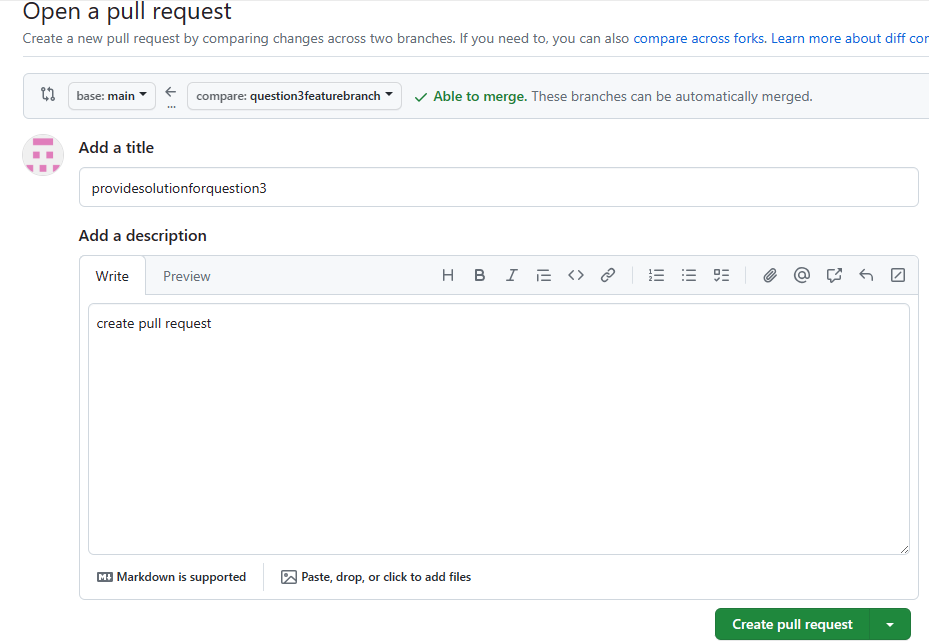
Description automatically generated

Select the base branch (usually "main") and the compare branch (your feature branch).

A screenshot of a computer

Description automatically generated

Review the changes and click on the "Create pull request" button to create the pull request.

A screenshot of a computer

Description automatically generated

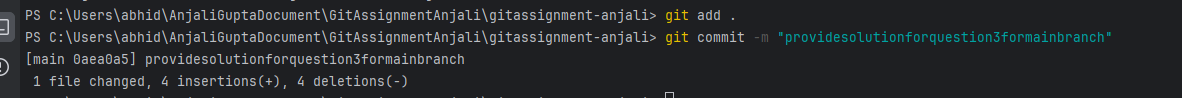
Step 6: As another user in the master branch make some changes to the same file.

Another user (or me on main branch) makes changes to the same file in the main branch.

A screenshot of a computer program

Description automatically generated

Step 7: Add and commit the changes to the main branch.



Step 8: Push the changes to the master branch.

A screen shot of a computer program

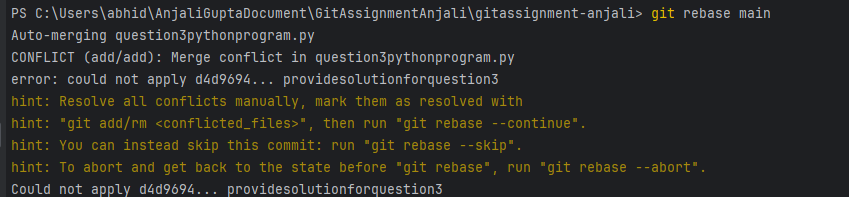
Description automatically generated

Note: There will be a conflict in the pull request, how do we resolve it??

A screenshot of a computer

Description automatically generated

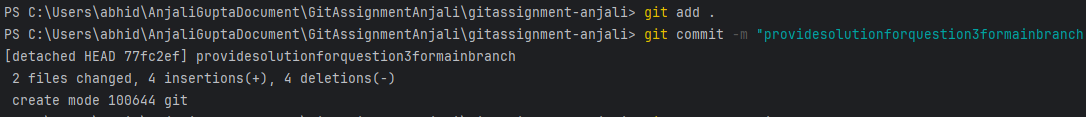
Step 9: Resolve the conflict in the pull request of feature branch.



A screenshot of a computer

Description automatically generated

Hint: git rebase



Type command ‘ git mergetool’ for seeing changes and take action according to file and resolve conflict

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Question 4**

Step 1: Step 1: Create a feature branch.



Step 2: Switch to the new branch.

open the file and make some changes to it.

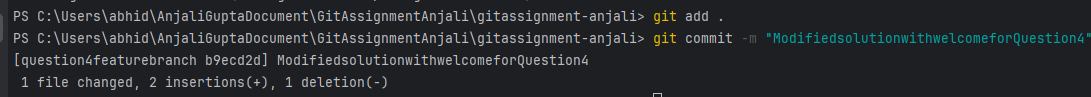
Add and commit the changes to the new branch.

A blurry image of text on a black background

Description automatically generated

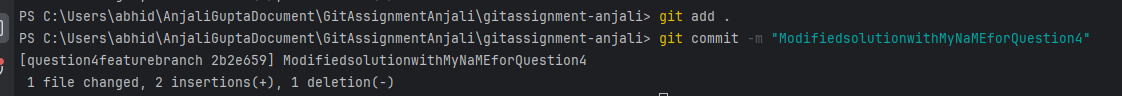
open the same file and make some changes to it.

Add and commit the changes to the new branch.



open the same file and make some changes to it.

Add and commit the changes to the new branch.

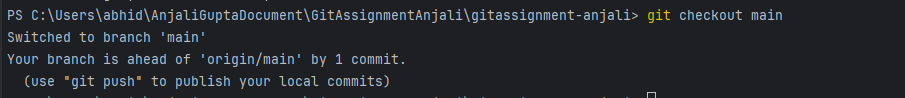


Step 3: Identify the commit or commits that you want to "cherry-pick"(Note the hash of the commit or commits that you want to "cherry-pick")

A screen shot of a computer

Description automatically generated

Step 4: Use the "git checkout" command to switch to the branch where you want to apply the changes.



Step 5: Use the "git cherry-pick" command followed by the commit hash(es) that you want to apply.

A black screen with white text

Description automatically generated A black screen with white text

Description automatically generated

**Question 5**

Step 1: Step 1: Create a feature branch.

A computer screen with text

Description automatically generated

Step 2: Switch to the new branch.

open the file and make some changes to it.

Add and commit the changes to the new branch.

A screen shot of a computer program

Description automatically generated

open the same file and make some changes to it.

Add and commit the changes to the new branch.

A screenshot of a computer program

Description automatically generated

open the same file and make some changes to it.

Add and commit the changes to the new branch.

A screenshot of a computer

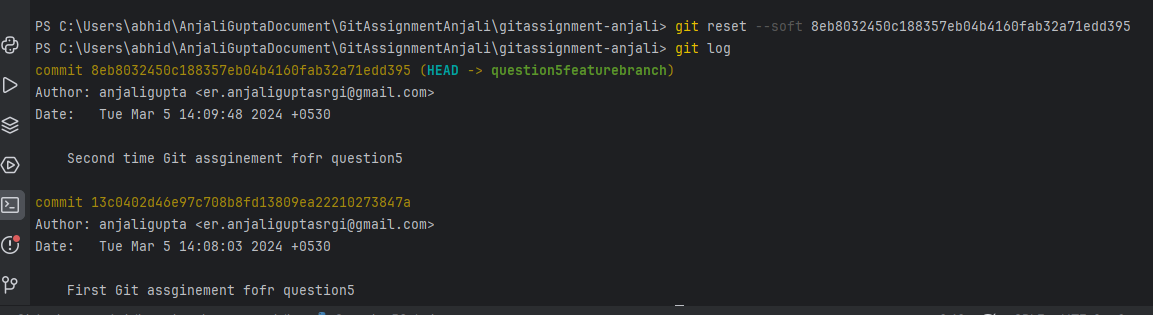
Description automatically generated

Step 3: Use the "git log" command to view the commit history and identify the commit to which you want to reset.

A computer screen shot of a computer code

Description automatically generated

Step 4: Use the "git reset" command followed by the desired reset type and the commit hash

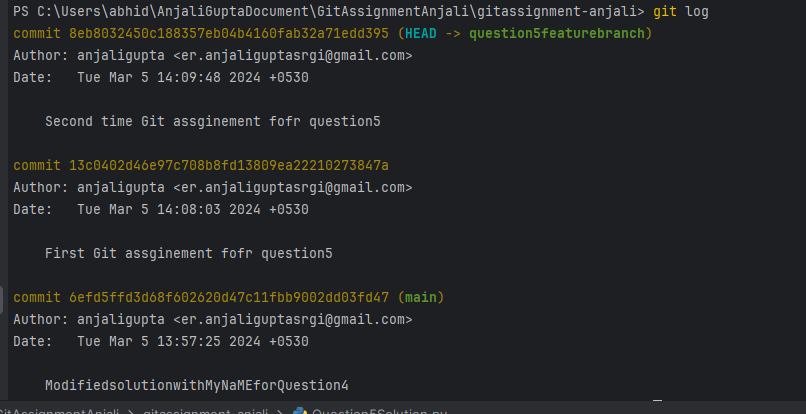


Step 5: Verify that the reset was successful by using the "git log" command again.

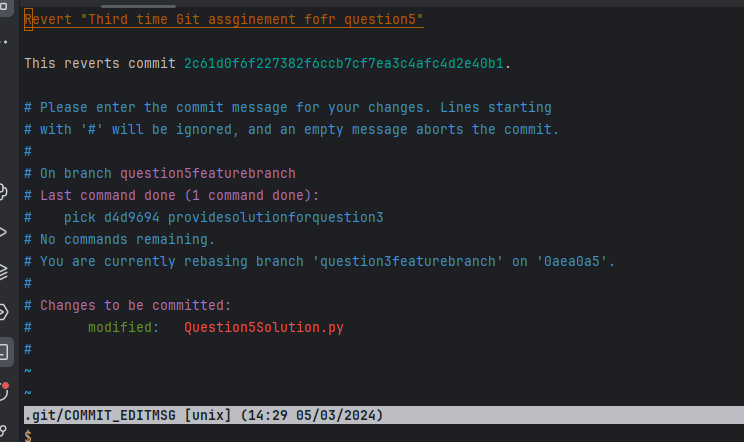
A screen shot of a computer

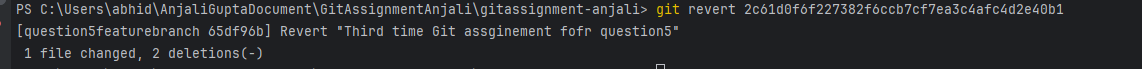
Description automatically generated

Step 6: Use the "git log" command to view the commit history and identify the commit that you want to reverse.



Step 7: Use the "git revert" command followed by the commit hash or reference to which you want to revert. (Hint: git revert <commit hash>)





Step 8: Verify that the revert was successful by using the "git log" command again.

A screen shot of a computer

Description automatically generated

Note: Identify the difference between git log after git reset and git r evert.

git reset:

When we use git reset to move the branch pointer to a specific commit, it effectively rewrites the commit history.

After using git reset, the commits between the original HEAD (where the branch pointer was before the reset) and the commit we reset to are effectively removed from the branch's history.

If we inspect the commit history with git log after using git reset, we'll see that the commits after the reset point are no longer present in the history.

git revert:

When we use git revert to revert a specific commit, Git creates a new commit that undoes the changes introduced by the specified commit.

After using git revert, the original commit remains in the history, and Git adds a new commit that negates the changes introduced by the reverted commit.

If you inspect the commit history with git log after using git revert, you'll see both the original commit and the revert commit in the history.

In summary, git reset changes the commit history by removing commits, while git revert keeps the commit history intact by adding new commits that reverse the changes introduced by specific commits.