**Assignment 1& 2 & 3**

**Assignment 1: Initialize a new Git repository in a directory of your choice. Add a simple In40 text file to the repository and make the first commit.**

**Ans:**

**1. Choose a Directory:**

Think of a location on your computer where you'd like to keep your project files. This could be your Documents folder, a specific project folder, or anywhere you like to organize your work.

**2. Open Terminal/Command Prompt:**

Navigate to the chosen directory using the cd command in your terminal/command prompt. For example, if your directory is on your Desktop, you might type:

cd Desktop

**3. Initialize Git Repository:**

Use the git init command in your terminal to create a new Git repository in the current directory. This will create a hidden folder named .git to store version control information.

git init

**4. Create and Add In40 Text File:**

You can create a new text file named "in40.txt" using your preferred text editor. Here's an example of some content you can add:

This is a simple In40 text file for Assignment 1.

Save the file within your current directory.

In your terminal, use the git add in40.txt command to add the file to the Git staging area. This tells Git that you want to include this file in the next commit.

**5. Commit the Changes:**

Use the git commit -m "Initial commit with In40 text file" command to create a snapshot of the current project state. The -m flag specifies a message describing the commit.

**Explanation:**

git commit: This command tells Git to create a commit.

-m "Initial commit with In40 text file": This part provides a descriptive message for your commit. It's important to include clear messages as you make more commits in the future.

Congratulations! You've successfully initialized a new Git repository, added an In40 text file, and made your first commit.

**Assignment 2: Branch Creation and Switching**

**Create a new branch named 'feature' and switch to it. Make changes in the 'feature' branch and commit them.**

**Ans:**

**1. Verify Current Branch (Optional):**

You can use the git branch command to see which branch you're currently on. Ideally, this should be the master branch created in Assignment 1.

**2. Create and Switch to 'feature' Branch:**

Use the following command to create a new branch named 'feature' and switch to it simultaneously:

git checkout -b feature

**Explanation:**

git checkout: This command is used to switch branches.

-b feature: The -b flag tells Git to create a new branch named 'feature' if it doesn't already exist and then switch to that branch.

**3. Make Changes in 'feature' Branch:**

Now that you're in the 'feature' branch, you can edit your "in40.txt" file or create new files specific to your feature development.

**4. Commit Changes in 'feature' Branch:**

Once you've made your changes, stage them for commit using git add <filename>. Here, <filename> could be "in40.txt" if you modified it, or the name of any new files you created.

Finally, commit your staged changes with a descriptive message using git commit -m "Your commit message here".

**Example:**

Let's say you added a new line to "in40.txt" and want to commit it:

git add in40.txt

git commit -m "Added a new line to in40.txt in feature branch"

Congratulations! You've created a new branch named 'feature', switched to it, made changes, and committed them to the 'feature' branch. This allows you to develop your feature independently without affecting the main project code (usually in the 'master' branch).

**Assignment 3: Feature Branches and Hotfixes**

**Create a 'hotfix' branch to fix an issue in the main code. Merge the 'hotfix' branch into 'main' ensuring that the issue is resolved.**

**Ans:**

**1. Identify the Issue and Branch from 'master':**

Use the following command to create a new branch named 'hotfix' from the current 'master' branch:

git checkout -b hotfix

This creates a new 'hotfix' branch that diverges from the exact state of the 'master' branch.

**2. Fix the Issue in 'hotfix' Branch:**

Switch to the 'hotfix' branch using:

git checkout hotfix

Now, edit the relevant files (like "in40.txt" in this example) to fix the issue.

**3. Commit the Fix in 'hotfix' Branch:**

Once you've fixed the bug, stage your changes using git add <filename>.

Commit your fix with a clear message using git commit -m "Fixed issue in <filename> (hotfix)".

**4. Merge 'hotfix' Branch into 'master':**

Switch back to the 'master' branch using:

git checkout master

Now, merge the 'hotfix' branch into 'master' to integrate the fix. Use the following command:

git merge hotfix

Resolving Merge Conflicts (if any):

In some cases, there might be merge conflicts if you or someone else made changes to the same part of the code in the 'master' branch since you created the 'hotfix' branch. Git will signal this and highlight the conflicting parts in the files.

You'll need to manually edit the files to resolve these conflicts and then add and commit the resolved files.

**5. Verify the Fix (Optional):**

You can switch back to the 'master' branch and build/test your code to ensure the hotfix resolved the issue.

**6. Push Changes (Optional):**

If you're working with a remote Git repository, you can push your changes to the remote server after a successful merge:

git push origin master

Congratulations! You've created a hotfix branch, fixed the issue, merged it into the 'master' branch, and (optionally) pushed the changes to the remote repository. This process helps isolate and quickly deploy bug fixes without affecting ongoing development in the 'master' branch.