Assignment 5.1

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

Here are the steps to initialize a new Git repository, add a simple text file, and make the first commit:

- Open your terminal or command prompt.
- Navigate to the directory where you want to initialize the Git repository using the cd command.

  For example: cd path/to/your/directory
- Initialize a new Git repository using the git init command:

  >git init
- Create a simple text file using a text editor or command-line tools.
  For example, you can use the echo command to create a text file named sample.txt:

  echo "This is a simple text file." >sample.txt
- Add the text file to the Git repository using the git add command:

  git add sample.txt
- Commit the changes to the repository using the git commit command. This will open a text editor where you can enter a commit message:

  git commit

  If you prefer to provide the commit message directly in the command line, you can use the -m flag:

  git commit -m "Initial commit: Added sample.txt"

Assignment 5.2

Given: Branch Creation and Switching: Create a new branch named 'feature' and switch to it. Make changes in the 'feature' branch and commit them.

Here are the steps to create a new branch named 'feature', switch to it, make changes, and commit them:

- First, ensure you are in the main branch (typically named 'master' or 'main'). You can switch to the main branch using the following command:

  git checkout main

- Create a new branch named 'feature' and switch to it using the following command:

  git checkout -b feature

  This command creates a new branch named 'feature' and switches to it in one step.

- Make the desired changes to your files. For example, let's add some text to thesample.txt' file:

  echo "This is a feature branch change." >>sample.txt

- Add the modified files to the staging area:

  git add sample.txt

- Commit the changes to the 'feature' branch with a meaningful commit message:

  git commit -m "Added feature branch change to sample.txt"

- Now, you've successfully created a new branch named 'feature', switched to it, made changes, and committed them.
- You can continue working in the 'feature' branch, making additional changes, and committing them as needed. To switch back to the main branch, you can use the command:

  git checkout main

Assignment 5.3

Given: 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.

Here's how you can create a 'hotfix' branch to fix an issue in the main code and then merge the 'hotfix' branch into 'main':

- First, ensure you are in the main branch. You can switch to the main branch using the following command:

  git checkout main

- Create a new branch named 'hotfix' to work on the issue:

  git checkout -b hotfix

- Make the necessary changes to fix the issue in your code.
- Add the modified files to the staging area:

  git add .
- Commit the changes to the 'hotfix' branch with a meaningful commit message:

  git commit -m "Fixed issue in main code"
- Now, you've successfully fixed the issue in the 'hotfix' branch.
- Switch back to the main branch:

  git checkout main

- Merge the 'hotfix' branch into the 'main' branch:

  git merge hotfix

- Resolve any merge conflicts if they occur. Git will prompt you to review and resolve conflicts manually.
- Once the merge is successful and the issue is resolved, commit the changes to the 'main' branch:

  git commit -m "Merged hotfix branch to main: Resolved issue"

- Finally, push the changes to the remote repository:

  git push origin main

- Now, the 'hotfix' branch has been merged into the 'main' branch, and the issue in the main code has been resolved.