Questions on Bash (Bash Practice Activity):

Complete the following tasks using Bash script in your terminal or GitBash:

1. Use the terminal/GitBash to navigate to the “Evangadi” folder located on your Desktop.
2. Inside your “Evangadi” folder, create a folder called “BashPractice”.
3. In the “BashPractice” folder, create another subfolder named “BashTest”.
4. In the “BashPractice” folder, create a text file named “myTextFile.txt”.
5. Add the text "Hello bash" to the “myTextFile.txt” file using the "echo" command.
6. Rename the “myTextFile.txt” file to “myTextPractice.txt”.
7. Change the permission of the “myTextPractice.txt” file to 776 using the chmod command.
8. Delete the “BashTest” folder along with any file inside it.

Question 2) Group Activity on Git

1. **Setting up a Repository, Cloning, and Collaborating**
   a. If you do not already have a GitHub account, create one using the following link: GitHub - Join.
   b. Arrange yourselves into groups of five and select one person to be the leader for this task. The group leader should create a new repository called **appleClone** in their GitHub account.
   c. The group leader should:
      - Set up the initial folder structure with the necessary files inside. (eg., an index.html file with boilerplate and style.css file)
      - Create an images folder, and store all the required images inside it. Make sure to set this up before pushing to GitHub.
      - Push this initial setup to the repository, allowing others to collaborate easily.
   d. The group leader should invite all group members as collaborators to the appleClone repository on GitHub.
      - Each member should accept the invitation email to join the repository.

2. **Cloning the Repository:**
   - Each group member should clone the appleClone repository to their local machine using the following terminal command (inside their Evangadi folder):

```
git clone https://github.com/<leader_username>/appleClone.git
```

3. **Working on Your Assigned Task**
   a. Each group member must create a branch with their assigned task name or their personal name for clarity. The branches should reflect the following tasks:
      - **Header** (e.g., branch name: header)
      - **Footer** (footer)
      - **First section banner** (16-inch MacBook Pro)
      - **Second section banner** (iPhone 11 Pro)

- **Third section banner** (iPhone 11)
- **Fourth section – left part** (Watch – Series 5)
- **Fourth section – right part** (Card is here)
- **Fifth section – left part** (tv+)
- **Fifth section – right part** (AirPods Pro)
- **Sixth section – left part** (MacBook Pro)
- **Sixth section – right part** (The new iPad)

b. Work on your assigned task by modifying the relevant part of the project.
   - **Implement your task on your own branch.**
   - **Make frequent commits with descriptive messages about the changes you've made.**
   - **Make a frequent staging (git add) and committing (git commit)**
   - **Example commit message:**

```
git add . or git add <your specific targeted content name to be staged>
 git commit -m " header section with navigation links task competed"
```

c. After completing your task and testing it locally, pull the latest changes from the remote repository before pushing your changes to avoid conflicts. use the following commands:

```
Git pull origin main
Git push -u origin <your branch-name> or git push origin <your_branch_name>
```

4. Code Review and Merging Process

a. Create a Pull Request (PR) for your branch and request a review from the group leader. This can be done on GitHub under the "Pull Requests" tab.

b. The group leader, along with other group members, should review the changes in each pull request.
   - **Discuss and resolve any conflicts that may arise.**
   - **Suggest improvements if necessary.**

c. Once everyone agrees on the changes, the group leader merges the branch into the main branch.

5. Additional Questions/Tasks for Collaboration:

- **Branch Naming Conventions:** Discuss with the group why it's important to use clear and descriptive branch names.
- **Conflicts:** What happens when two group members modify the same file? How would you resolve merge conflicts within your group?

- **Reverting Changes:** If something breaks after merging a pull request, how would you revert back to a stable version of the project?
- **Best Practices:** Discuss the importance of small and frequent commits versus large and single commits.
- **Documentation:** Work together as a group to create a README.md file that describes your project, how to set it up locally, and any special instructions.

Note:

This structure encourages group collaboration, communication, and learning best practices for Git. Each student gets hands-on experience, while the group leader manages the overall workflow and conflict resolution.