Initial Setup

Task-1: Initialize a Git repository

1. Navigate to your project directory.
2. Run the command:

```
git init
```

This initializes a Git repository in your project folder.

Task-2: Create a file named script.js with a simple JavaScript function

1. Create a script.js file:

```
touch script.js
```

2. Add a simple function in script.js:

```
function sayHello() {
  console.log("Hello, world!");
}
```

Task-3: Add and commit the changes

1. Add the file to staging:

```
git add script.js
```

2. Commit the changes:

```
git commit -m "Add simple sayHello function"
```

Branching and Feature Development

Task-1: Create a new branch named feature-greeting

```
git checkout -b feature-greeting
```

Task-2: Switch to the new branch

(Already done when creating the branch with -b in the previous step.)

Task-3: Add a personalized greeting function in script.js

1. Modify script.js:

```
function personalizedGreeting(name) {
  console.log(`Hello, ${name}!`);
}
```

Task-4: Commit the changes in the feature branch

1. Stage and commit the changes:

```
git add script.js
git commit -m "Add personalizedGreeting function"
```

Task-5: Merge feature-greeting into the main branch

1. Switch to the main branch:

```
git checkout main
```

2. Merge the feature branch:

git merge feature-greeting

Task-6: Switch back to the main branch

(Already done when merging.)

Push to GitHub and Collaborate

Task-1: Create a GitHub repository

1. Go to GitHub and create a new repository.

Task-2: Push your local repository to GitHub

1. Add the remote repository:

git remote add origin https://github.com/your-username/your-repo-name.git

2. Push the code:

git push -u origin main

Task-3: Invite a team member to collaborate

1. Go to your GitHub repository settings and add a collaborator by their GitHub username or email.

Task-4: Team member clones the repository

Your team member runs:

git clone https://github.com/your-username/your-repo-name.git

Task-5: Fetch and merge the changes made by the team member

1. Fetch changes from GitHub:

git fetch origin

2. Merge their changes:

git merge origin/main

Resolve Merge Conflicts

Task-1: Create a new branch named feature-update

git checkout -b feature-update

Task-2: Make changes to script.js in feature-update branch

1. Modify script.js:

```
function updatedFeature() {
  console.log("This is the updated feature.");
}
```

2. Commit the changes:

git add script.js

git commit -m "Add updatedFeature function"

Task-3: Switch back to the main branch and make different changes

1. Switch to the main branch:

git checkout main

2. Modify script.js differently:

```
function mainUpdate() {
  console.log("This is an update in the main branch.");
}
```

3. Commit the changes:

```
git add script.js
git commit -m "Update script.js in the main branch"
```

Task-4: Attempt to merge feature-update into the main branch

1. Try merging:

```
git merge feature-update
```

If there's a conflict, Git will indicate the conflicting sections in script.js.

Task-5: Resolve any merge conflicts

1. Open script.js and manually resolve the conflicts by editing the code.
2. After resolving, stage the changes:

```
git add script.js
```

3. Complete the merge:

```
git commit
```

Undoing Changes

Task-1: Create a new file named obsolete.js

1. Create the file:

```
touch obsolete.js
```

Task-2: Add and commit the file

1. Stage and commit:

```
git add obsolete.js
git commit -m "Add obsolete.js"
```

Task-3: Remove the unnecessary obsolete.js file

1. Remove the file:

```
git rm obsolete.js
```

2. Commit the removal:

```
git commit -m "Remove obsolete.js"
```

Task-4: Undo the last commit

1. To undo the commit while keeping the file in the working directory:

```
git reset --soft HEAD~1
```