Problem Statement 1:

New to git? Follow the steps below to get comfortable making changes to the code base, opening up a pull request (PR), and merging code into the primary branch. Any important git and GitHub terms are in bold with links to the official git reference materials.

Follow the following consideration while solving the below problem.

1. Install git and create a GitHub account.
2. Create a local git repository.
3. Add a new file to the repo.
4. Add a file to the staging environment.
5. Create a commit.
6. Create a new branch.
7. Create a new repository on GitHub.
8. Push a branch to GitHub.
9. makes a temporary, local save of your code.
10. let's you tidy up your code before doing a commit.
11. Use the function that allows you to hunt out bad commits.
12. commits the combine squash in git.
13. Apply changes from one branch onto another.

Solution:

1. Install Git and create a GitHub account:

To install Git on your computer, you can follow the steps appropriate for your operating system. Below are instructions for common operating systems:

1. Download the Git installer for Windows from the official Git website: **Git for Windows**
2. Run the downloaded installer.
3. Follow the on-screen instructions in the installer. You can generally leave the default settings as they are unless you have specific preferences.
4. During the installation, you will reach a screen titled "Adjusting your PATH environment." Choose the default option "Use Git from the Windows

Command Prompt" or select the other option if you prefer using Git from the Git Bash terminal.

5. Continue with the installation, and once it's completed, click "Finish."

6. Open a new command prompt or Git Bash terminal and verify the installation by typing:

```
C:\Users\DELL>git --version
git version 2.39.2.windows.1
```

Steps to create a github account :

Visit the GitHub website:

1. Go to **GitHub** in your web browser.
2. Sign Up:
    a. On the GitHub homepage, you'll find a "**Sign Up**" button. Click on it.
3. Provide Your Information:
    a. Fill in the required information on the Sign Up page:
        i. **Username:** Choose a unique username for your GitHub account.
        ii. **Email address:** Enter your email address.
        iii. **Password:** Create a strong password for your account.
4. Complete the CAPTCHA:
    a. Complete any **CAPTCHA** or security checks to verify that you're a human.
5. Choose Your Plan:
    a. GitHub offers both free and paid plans. For most users, the free plan is sufficient. Choose the plan that suits your needs.
6. Verify Your Email:
    a. After completing the sign-up process, GitHub will send you an email to verify your email address. Open your email and click on the verification link provided by GitHub.

7. Welcome to GitHub:
    a. Once your email is verified, you'll be directed to your GitHub dashboard.

2. Create a local git repository:

Configure git and create a folder named demo git and change the directory to the demo git folder and initialize a repo:

```
C:\Users\DELL>git config --global user.name "kavin"

C:\Users\DELL>git config --global user.email "gkavinvelavan@gmail.com"

C:\Users\DELL>cd demo git

C:\Users\DELL\demo git>git init
Reinitialized existing Git repository in C:/Users/DELL/demo git/.git/
```

3. Add a new file to the repo:
Create a new file named newfile.txt in the demo git folder
touch newfile.txt

4. Add a file to the staging environment :

```
C:\Users\DELL\demo git>git add newfile.txt

C:\Users\DELL\demo git>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   newfile.txt
```

5. Create a commit

```
C:\Users\DELL\demo git>git commit -m "new file"
[master (root-commit) 6993078] new file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newfile.txt

C:\Users\DELL\demo git>git status
On branch master
nothing to commit, working tree clean
```

6. Create a new branch:

```
C:\Users\DELL\demo git>git branch newbranch

C:\Users\DELL\demo git>git branch
* master
  newbranch
```

7. Create a new repository on GitHub:

Step 1 : login to github account with valid user name and password
Step 2 : create a new repo on clicking + symbol and name a repo demogit
Step 3 : click finish

8. Push a branch to GitHub:

Switch to new branch

```
C:\Users\DELL\demo git>git checkout newbranch
Switched to branch 'newbranch'

C:\Users\DELL\demo git>git branch
  master
* newbranch
```

```
C:\Users\DELL\demo git>git remote add origin https://github.com/kavinvelavan/demogit.git

C:\Users\DELL\demo git>git push -u origin newbranch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 211 bytes | 105.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kavinvelavan/demogit.git
 * [new branch]      newbranch -> newbranch
branch 'newbranch' set up to track 'origin/newbranch'.
```

Pull is a combination of fetch and merge. It is used to pull all changes from a remote repository into the branch you are working on.

```
C:\Users\DELL\demo git>git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/kavinvelavan/demogit/pull/new/master
remote:
To https://github.com/kavinvelavan/demogit.git
 * [new branch]      master -> master

C:\Users\DELL\demo git>git pull origin master
From https://github.com/kavinvelavan/demogit
 * branch            master     -> FETCH_HEAD
Already up to date.

C:\Users\DELL\demo git>git branch -d newbranch
Deleted branch newbranch (was 6993078).
```

9. makes a temporary, local save of your code.

```
C:\Users\DELL\demo git>git checkout new
Switched to branch 'new'

C:\Users\DELL\demo git>git add .

C:\Users\DELL\demo git>git commit -m "Temp file"
On branch new
nothing to commit, working tree clean
```

10. let's you tidy up your code before doing a commit.

```
C:\Users\DELL\demo git>git add .

C:\Users\DELL\demo git>git status
On branch checkout
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   newfile.txt
```

11. Use the function that allows you to hunt out bad commits.

```
C:\Users\DELL\demo git>git bisect start
status: waiting for both good and bad commits

C:\Users\DELL\demo git>git bisect bad
status: waiting for good commit(s), bad commit known

C:\Users\DELL\demo git>git bisect good
6993078a188f86cb045b682abefb3bba7a2dd47e was both good and bad
```

12. commits the combine squash in git.

```
C:\Users\DELL\demo git>git status
On branch checkout
No commands done.
Next command to do (1 remaining command):
   pick 71b4269 changed
  (use "git rebase --edit-todo" to view and edit)
You are currently editing a commit while rebasing branch 'checkout' on '6993078'.
  (use "git commit --amend" to amend the current commit)
  (use "git rebase --continue" once you are satisfied with your changes)

nothing to commit, working tree clean

C:\Users\DELL\demo git>git rebase --continue
On branch checkout
Last command done (1 command done):
   pick 71b4269 changed
No commands remaining.
You are currently rebasing branch 'checkout' on '6993078'.
  (all conflicts fixed: run "git rebase --continue")
```

13. Apply changes from one branch onto another.

Checkout the Target Branch:

First, make sure you are on the branch where you want to apply changes

git checkout target-branch

Merge the Source Branch:

Use git merge to merge changes from the source branch into the target branch.

git merge source-branch

Commit the Merge:

After resolving conflicts (if any), commit the merge.

git commit -m "Merge changes from source-branch"